Floating Point Applications On FPGAs Can Be Competitive*

Martin Herbordt     Bharat Sukhwani
Matt Chiu     Ashfaq Khan

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

*This work is supported in part by the U.S. NIH/NCRR and by IBM
In the end all architectures, CPU, GPU, FPGA, etc. will converge ....

-- Pradeep Dubey (Intel), SAAHPC09
In the end all architectures, CPU, GPU, FPGA, etc. will converge ....
But we need to have really fast scatter/gather and synchronization ....

-- Pradeep Dubey (Intel), SAAHPC09
In the end all architectures, CPU, GPU, etc. will converge ....
But we need to have really fast scatter/gather and synchronization ....

-- Pradeep Dubey (Intel), SAAHPC09

More active transistors, higher frequency (2003)
More active transistors, higher frequency (2007)
More active transistors, higher frequency (2012)
More active transistors, higher frequency

-- Brian Flachs (IBM), SAAHPC 2009
In the end all architectures, CPU, GPU, etc. will converge ....
But we need to have really fast scatter/gather and synchronization ....

-- Pradeep Dubey (Intel), SAAHPC09

More active transistors, higher frequency  (2003)
More active transistors, higher frequency  (2007)
More active transistors, higher frequency  (2012)
More active transistors, higher frequency

In the end the only way to get better performance is with special purpose hardware ....

-- Brian Flachs (IBM), SAAHPC 2009
In the end all architectures, CPU, GPU, etc. will converge ....
But we need to have really fast scatter/gather and synchronization ....

-- Pradeep Dubey (Intel), SAAHPC09

More active transistors, higher frequency  (2003)
More active transistors, higher frequency  (2007)
More active transistors, higher frequency  (2012)

In the end the only way to get better performance is with special purpose hardware ....

-- Brian Flachs (IBM), SAAHPC 2009

*In the end we’re all dead ....*

-- John Maynard Keynes
Theme of this talk …

FPGAs give you

- Application-specific processors
- Single cycle scatter/gather & synchronization
- In general, lot’s of degrees of freedom
Peak Performance in FLOPs – i7

2/4 FMUL, 2/4 FADD per cycle
4 DP or 8 SP per cycle per core
i7 = 4 cores, 3GHz

Peak Performance =
48 GFLOPs (DP)
96 GFLOPs (SP)

FP for FPGAs
Peak Performance in FLOPs – Tesla 1060

SP core → 1 FMUL, 1 FMAD per cycle for 3 FLOPs / cycle / SP core
SM → 8 SP cores       Tesla 1060 → 30 SMs @ 1.3GHz
Peak Performance = 936 GFLOPs (SP)
Peak Performance in FLOPs – FPGAs (65nm)

• **72 GFLOPs (SP)**
  – Stratix-III 340 → 144 FMUL, 215 FADD @ 200MHz
  – mix from a systolic array for a multi-parameter correlation
  – conservative for an application with short interconnects
  – Uses Altera IP cores

• **190 GFLOPs (SP)**
  – Virtex-5 SX240 (www.xilinx.com)
  – Very high clock frequency? Custom FP blocks?

• **85-94 GFLOPs (SP)**
  – Virtex-5 LX330 (Strenski, 2009)
  – various very conservative assumptions
  – not real applications
Key question …

How realistic is peak performance?
Utilization Examples – Core 2 Quadcore

1. 128³ point 3D FFT
   – FFTW (perhaps not the ultimate, but widely used)
   – Auto-tuned
   – 58ms
   – 1.6 GFLOPs (DP) on four 3 GHz cores or ~8% of peak
   – Higher utilizations have been achieved

2. Matrix-matrix multiply -- with icc -O2 with blocking
   – various reasonable matrix and block sizes
   – compiler vectorizes and unrolls
   – 1.77 GFLOPs on one 2 GHz core or ~22% of peak

3. Matrix-matrix multiply -- maximally tuned
   – Very high utilizations have been achieved ~90% of peak
Utilization Examples – Tesla 1060

• $128^3$ 3D FFT
  – Nvidia CUFFT library function
  – 6.3 ms (not including transfer time) for ~ 4% of peak

• Matrix-matrix multiply
  – 60% or more of peak has been achieved
Why is FP on FPGAs plausible?

With FPGA’s flexibility (additional degrees of freedom):

- Can improve raw capability
- Can achieve high utilization

As always with FPGAs,

Flexibility $\rightarrow$

- reduced chip area per computation $\rightarrow$
- additional parallelism $\rightarrow$
- improved performance
Improving Raw FP Capability of FPGAs

1. FP Compiler (Langhammer et al.)
   - Optimize FP pipelines

2. Application-specific arithmetic
   - Reduced precision
   - Fixed point
   - Hybrid representations
   - ...

FP for FPGAs

HPEC 2009
Altera Floating Point Compiler (FPC)

• In a datapath, some redundancy can be removed, especially within a cluster of similar or identical operations

• The advantage of FPC is achieved by
  – Removal of redundant normalizations within a cluster
    • Keep un-normalization with large precision, if possible
    • Normalization usually are performed out of a local cluster or out of datapath
  – Change in the internal number representation and format

• The efficiency of FPC*
  – Up to 50% reduction in soft logic, no gain for hard-multipliers; leaving more space for routing optimization and frequency improvement
  – Up to 50% of latency saving
  – Linear-algebra applications with 1:1 adder/multiplier ratio
    • 100 GFLOPs was reported for xGEMM application, running at 250MHz Stratix EP3SE260

*Altera RSSI 2008, Martin Langhammer
# Application Specific Arithmetic

## Fixed-Point Addition

<table>
<thead>
<tr>
<th>Precision</th>
<th>ALUT</th>
<th>DSP (18 x 18)</th>
<th>Latency</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>0.009%</td>
<td>0%</td>
<td>2</td>
</tr>
<tr>
<td>16</td>
<td>0.017%</td>
<td>0%</td>
<td>2</td>
</tr>
<tr>
<td>32</td>
<td>0.032%</td>
<td>0%</td>
<td>2</td>
</tr>
<tr>
<td>64</td>
<td>0.064%</td>
<td>0%</td>
<td>2</td>
</tr>
</tbody>
</table>

## Single Precision FP*

<table>
<thead>
<tr>
<th>Core</th>
<th>ALUT</th>
<th>DSP (18x18)</th>
<th>Latency</th>
</tr>
</thead>
<tbody>
<tr>
<td>Addition</td>
<td>0.25%</td>
<td>0%</td>
<td>10</td>
</tr>
<tr>
<td>Multiplier</td>
<td>0.13%</td>
<td>0.5%</td>
<td>8</td>
</tr>
<tr>
<td>Divider</td>
<td>0.18%</td>
<td>1.7%</td>
<td>15</td>
</tr>
<tr>
<td>Inverse Root</td>
<td>0.16%</td>
<td>1.4%</td>
<td>19</td>
</tr>
<tr>
<td>EXP</td>
<td>0.26%</td>
<td>1.4%</td>
<td>16</td>
</tr>
<tr>
<td>LOG</td>
<td>0.59%</td>
<td>0.9%</td>
<td>21</td>
</tr>
</tbody>
</table>

## Fixed-Point Multiplication

<table>
<thead>
<tr>
<th>Precision</th>
<th>ALUT</th>
<th>DSP (18 x 18)</th>
<th>Latency</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>0%</td>
<td>0.13%</td>
<td>3</td>
</tr>
<tr>
<td>16</td>
<td>0%</td>
<td>0.26%</td>
<td>3</td>
</tr>
<tr>
<td>32</td>
<td>0%</td>
<td>0.52%</td>
<td>3</td>
</tr>
<tr>
<td>64</td>
<td>0.12%</td>
<td>4.17%</td>
<td>4</td>
</tr>
</tbody>
</table>

## Double Precision FP*

<table>
<thead>
<tr>
<th>Core</th>
<th>ALUT</th>
<th>DSP (18x18)</th>
<th>Latency</th>
</tr>
</thead>
<tbody>
<tr>
<td>Addition</td>
<td>0.49%</td>
<td>0%</td>
<td>10-14</td>
</tr>
<tr>
<td>Multiplier</td>
<td>0.29%</td>
<td>1.3%</td>
<td>7</td>
</tr>
<tr>
<td>Divider</td>
<td>0.44%</td>
<td>3.5%</td>
<td>21</td>
</tr>
<tr>
<td>Inverse Root</td>
<td>0.44%</td>
<td>3.5%</td>
<td>32</td>
</tr>
<tr>
<td>EXP</td>
<td>0.81%</td>
<td>3.6%</td>
<td>22</td>
</tr>
<tr>
<td>LOG</td>
<td>1.01%</td>
<td>1.6%</td>
<td>26</td>
</tr>
</tbody>
</table>

*Resource utilization numbers are referred to Altera Stratix EP3SE260

*RSSI 2008, Martin Langhammer

FP for FPGAs  
HPEC 2009
Enabling Factors for High Utilization

1. Choice of FP units
   - Exactly those that you need

2. Ordering/positioning FP units
   - Pass data directly from one to the next

3. Embedding non-FP ops to reduce data movement
   - Constants, data-dependent coefficients

4. Control is embedded in the interconnect

5. Flexible communication helps keep pipelines filled
   - Complex patterns, scatter/gather (broadcast/reduce)
Molecular Dynamics

MD – An iterative application of Newtonian mechanics to ensembles of atoms and molecules

Runs in phases:

- Force update
- Motion update (Verlet)

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

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

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

\[ F_{i}^{C} = q_{i} \sum_{j \neq i} \left( \frac{q_{j}}{r_{ji}^3} \right) \vec{r}_{ji} \]

*IET/CDT06, ParC08, FPL09

HPEC 2009
Examine Short-Range Force Kernel

- Non-bonded short-range force consumes >90% of total simulation time

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

- LJ force
- short range component of Coulomb force
- smoothing function
Force Pipeline

1. Choice of FP units
   Exactly those that you need:
   16 FADD, 16FMUL, 1 FDIV, 1 FSQRT

2. Ordering/positioning FP units
   Pass data directly from one to the next

3. Embedding non-FP ops

4. Embedded control

5. Flexible communication helps keep pipelines filled
Force Pipeline 1: Double Precision

Double Precision

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>IP Core</td>
<td>double</td>
<td>21% (10% / 18%)</td>
<td>27%</td>
<td>181</td>
</tr>
</tbody>
</table>

Number of force pipelines = 3
Throughput = 540M forces/s

FP for FPGAs  HPEC 2009
Force Pipeline 2: Single Precision

Single precision

- Exclusion (small r) computed on host
- Precision may not be adequate

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>IP Core</td>
<td>double</td>
<td>21% (10% / 18%)</td>
<td>27%</td>
<td>181</td>
</tr>
<tr>
<td>IP Core</td>
<td>single</td>
<td>10% (5% / 9%)</td>
<td>11%</td>
<td>184</td>
</tr>
</tbody>
</table>

Number of force pipelines = 9
Throughput = 1.7G (3.15x DP)
Force Pipeline 3: Floating Point Compiler

Optional Fixed/FP Converter

\[ r^2 \]
\[ r^{-1} \]
\[ r^{-2} \]
\[ r^{-6} \]
\[ r^{-3} \]
\[ r^{-14} \]
\[ r^{-8} \]

FP for FPGAs HPEC 2009
Force Pipeline 3: Floating Point Compiler

Single precision + optimization w/ FPC
- Exclusion (small r) computed on host
- Floating point parts optimized with FPC

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>IP Core</td>
<td>double</td>
<td>21% (10% / 18%)</td>
<td>27%</td>
<td>181</td>
</tr>
<tr>
<td>IP Core</td>
<td>single</td>
<td>10% (5% / 9%)</td>
<td>11%</td>
<td>184</td>
</tr>
<tr>
<td>FPC*</td>
<td>single</td>
<td>7% (4% / 6%)</td>
<td>11%</td>
<td>202</td>
</tr>
</tbody>
</table>

Number of force pipelines = 9
Throughput = 1.8G  (3.3x DP)
Force Pipeline 4: Some Fixed Point

Hybrid: Single precision + 32 bit fixed point
- Exclusion (small r) computed on host
- 32-bit precision likely to be adequate (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>IP Core</td>
<td>double</td>
<td>21% (10% / 18%)</td>
<td>27%</td>
<td>181</td>
</tr>
<tr>
<td>IP Core</td>
<td>single</td>
<td>10% (5% / 9%)</td>
<td>11%</td>
<td>184</td>
</tr>
<tr>
<td>FPC*</td>
<td>single</td>
<td>7% (4% / 6%)</td>
<td>11%</td>
<td>202</td>
</tr>
<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 (5.2x DP)
O(N) with Cell Lists

Observation:
- Typical volume to be simulated = 100 Å³
- Typical LJ cut-off radius = 10 Å

Therefore, for all-to-all O(N²) 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. particles in adjacent cells.
- Issue → shape of cell – spherical would be more efficient, but cubic is easier to control
- Issue → size of cell – smaller cells mean less useless force computations, but more difficult control. Limit is where the cell is the atom itself.
Problem with Cell Lists

• Problem → Efficiency
  – ~15.5% efficiency when cell size is equal to cutoff
  – 84.5% of computation is wasted

• Solution → Filtering*
  – Remove unnecessary particles outside cutoff
    • on-the-fly neighbor lists
  – Only compute forces for “valid” particle pairs

*FPL 2009
Idea: $\rightarrow 6x$ as many “filters” as force pipelines
$\rightarrow$ Throughput improves $6x$ for marginal cost
More ideas -- reduce precision of filter -- approximate spherical geometry

<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 muls</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 muls</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>

FP for FPGAs
MD Design (so far in this talk)

For all particle pairs \((P_a, P_b)\)
- Small \(r\) gets done on host in DP
- \(r < \text{cut-off}\) gets done with force pipeline
- \((P_a, P_b)\) in cell-set gets done with filter
- \((P_a, P_b)\) not in cell-set computed on host \(\rightarrow O(1)\)

A high end FPGA (65nm) fits
- 8-10 force pipelines
- 80 filters
Much more to get this working ...

High Level Loop
- Cells are mapped to BRAMs
- Process one home cell at a time
- Load 9 new cells per home cell
Mapping Particles Pairs to Filters

- Particle Mapping (PM)
  - Each filter is responsible for a different reference particle
  - Each cycle, a single partner particle from the cell set is broadcast to all of the filters
  - Particle pairs which satisfy filtering rules are stored into filter queues for force computations

8-9 Filters / force pipeline

Reference particles are distributed into each filter

Neighboring cells

Home cell

Force Pipeline
Filtering List – Continuous Queueing

- “Matching” pairs generated by filters are directed into FIFOs and processed by force pipelines
- Concentrator is responsible to drain matching pairs from FIFOs to force pipeline
  - Round-Robin arbiter among queues with priority given first to those that are full and second to those that are non-empty
- Assert “stall” signal while more than one queue is almost full
- Less memory space required but more complex logic

Stall signal is asserted when more than one queue is almost full

Force Pipeline

Periodic Boundary Condition Full Precision Filtering Force Computations

Concentrator
Round-Robin with priority

Priority is given to a queue which is almost full
Queue Size

- Queue size of 8 is sufficient to achieve 99.2% high utilization
MD Results

• 8 force pipelines fit easily into a Altera Stratix III EP3SE260
  – Place & Route
  – Simulation quality validated with standard methods (see HPRCTA08)
  – Runs @ ~200 MHz
  – Each filter bank contains 8-9 filters and one force pipelines
  – Over 95% utilization (data transfers discussed elsewhere)

• Performance on short-range non-bonded force computation
  – 8 x 200MHz x 95% = 1.5G “payload” force computations per second
  – 18x performance of original DP implementation
  – Executes one iteration of standard 90K particle ApoA1 benchmark in 20 ms
  – 85x speed-up over single core
## Performance, cont.

<table>
<thead>
<tr>
<th></th>
<th>Performance X single core</th>
<th>System Power</th>
<th>Performance / Power (Normalized)</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>Quadcore</td>
<td>4</td>
<td>90W</td>
<td>1</td>
<td>$3K</td>
</tr>
<tr>
<td>GPU - Tesla 1060</td>
<td>25</td>
<td>250W</td>
<td>3.4</td>
<td>$4K</td>
</tr>
<tr>
<td>FPGA - Stratix-III</td>
<td>80</td>
<td>100W</td>
<td>18</td>
<td>$10K</td>
</tr>
<tr>
<td>ASIC - Desmond</td>
<td>1,280</td>
<td>?</td>
<td>?</td>
<td>$1M</td>
</tr>
</tbody>
</table>

FP for FPGAs | HPEC 2009
Why is Docking so important?

**Problem:** Combat the bird flu virus

**Method:** Inhibit its function by “gumming up” *Neuraminidase*, a surface protein, with an inhibitor

- *Neuraminidase helps release progeny viruses from the cell.*

**Procedure***:

- Search protein surface for likely sites
- Find a molecule that binds there (and only there)

*FPL 2008, GPGPU 2009
#From *New Scientist* [www.newscientist.com/channel/health/bird-flu](http://www.newscientist.com/channel/health/bird-flu)
Modeling Rigid Docking

Rigid-body approximation
Grid based computing
Exhaustive 6D search
Low Precision Data

Pose score = 3D correlation sum

\[ E(\alpha, \beta, \gamma) = \sum_{p} \sum_{i,j,k} R_{p}(i,j,k) \cdot L_{p}(i+\alpha, j+\beta, k+\gamma) \]

FFT to speedup the correlation

Reduces from \( O(N^6) \) to \( O(N^3 \log N) \)

Image courtesy of Structural Bioinformatics Lab, BU
Results

Speedup on different architectures

* Baseline: Best Correlation on single core

* Baseline: PIPER running on single core
Discrete Event Simulation of MD*

- Simulation with simplified models
- Approximate forces with barriers and square wells
- Classic discrete event simulation

*FPL07, FCCM08

FP for FPGAs

HPEC 2009
An Alternative ...

*Only update particle state when “something happens”*

- “Something happens” = a discrete event
- Advantage $\rightarrow$ DMD runs $10^5$ to $10^9$ times faster than tradition MD
- Disadvantage $\rightarrow$ Laws of physics are continuous
Discrete Event Simulation

Simulation proceeds as a series of discrete element-wise interactions

- **NOT** time-step driven

- Seen in simulations of ...
  - Circuits
  - Networks
  - Traffic
  - Systems Biology
  - Combat
PDES - Why is it hard for DMD?

Event propagation can be infinitely fast over any distance!

Note: a “chain” with rigid links is analogous and much more likely to occur in practice.
Overview - Dataflow

Main idea: DMD in one big pipeline

- Events processed with a throughput of one event per cycle
- Therefore, *in a single cycle*:
  - State is updated (event is committed)
  - Invalidations are processed
  - New events are inserted – up to four are possible

![Diagram](image-url)
On-Chip, “Scrunching” Priority Queue

Queue operation → Each cycle:
- Dequeue next event
- Up to four insertions
- Unbounded invalidates
- Fill holes with “scrunching” (conditional two-step advance)

Insertion/deletion priority queue

Queue element

FP for FPGAs  HPEC 2009
DMD Summary

Key Methods:
• Associative processing: broadcast, compare, etc.
• Standard HW components: priority queue, etc.

Performance –
• 200x – 400x for small to medium sized models
• 3D PDMD is difficult to scale to more than a small number of cores
Summary

• 2007-era FPGAs have similar raw FP capability as current high-end microprocessors
• FPGA flexibility gives numerous opportunities to substantially increase that FP capability
  – Selectable mode and precision
  – Custom pipelines
  – Pipeline-specific optimizations
• FPGAs can be configured to obtain extremely high utilization. Key capabilities include:
  – Single cycle communication and synchronization
  – Single cycle broadcast/reduce (scatter/gather)
  – Flexible communication
• Demonstration on high-impact applications
Questions?