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

BOSTON UNIVERSITY

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)
~~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 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

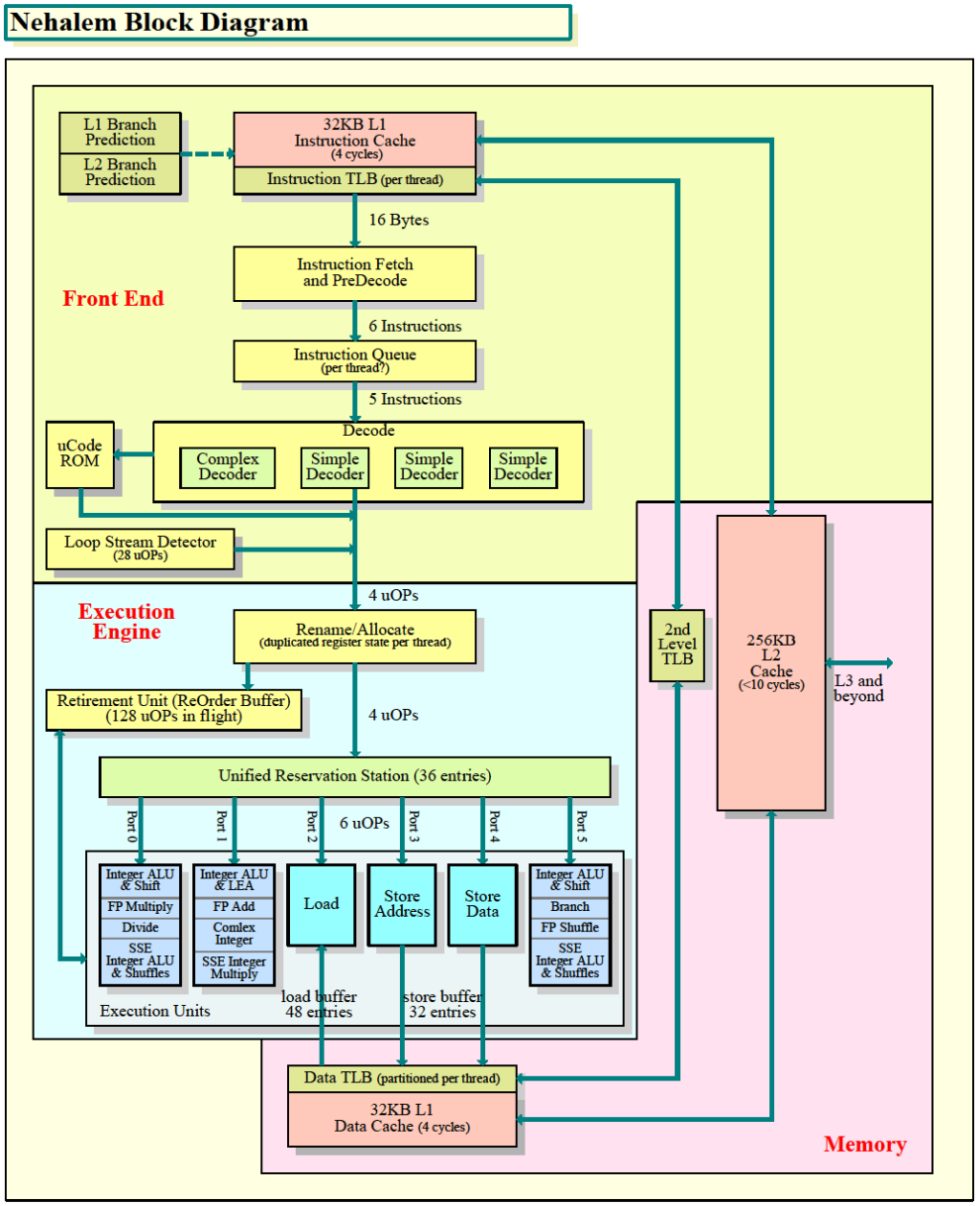
4 DP or 8 SP per cycle per core

i7 = 4 cores, 3GHz

Peak Performance =

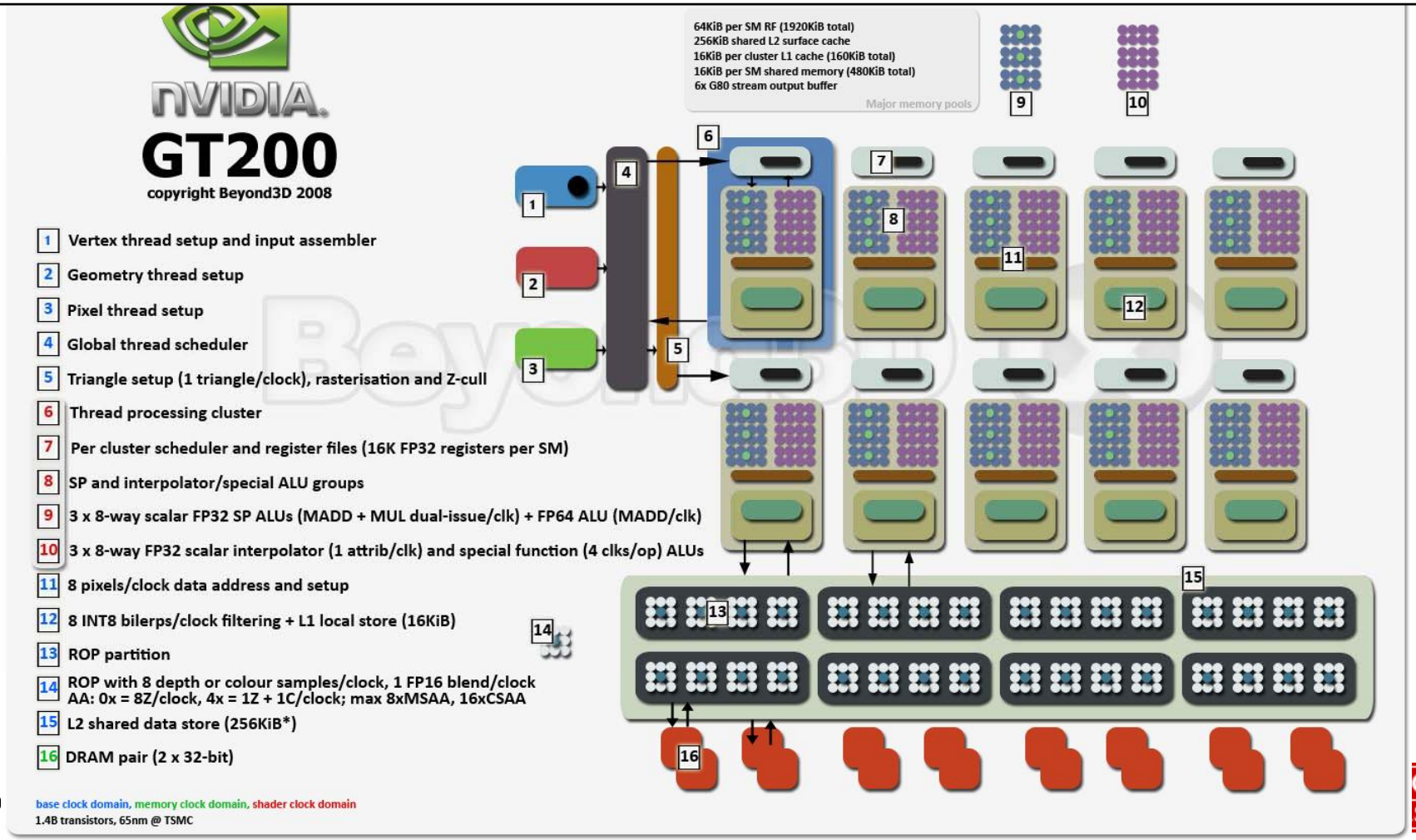
48 GFLOPs (DP)

96 GFLOPs (SP)



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)**



FP f

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. **128^3 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 →

reduced chip area per computation →

additional parallelism →

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
   - …

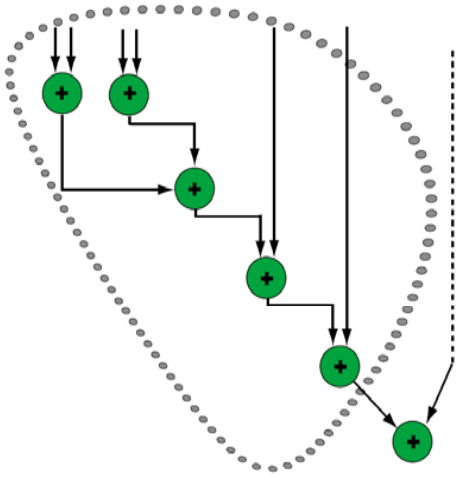
Altera Floating Point Compiler (FPC)

- 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

Cluster of identical or similar operations



Courtesy Altera

*Altera RSSI 2008, Martin Langhammer

Application Specific Arithmetic

Fixed-Point Addition

Precision	ALUT	DSP (18 x 18)	Latency
8	0.009%	0%	2
16	0.017%	0%	2
32	0.032%	0%	2
64	0.064%	0%	2

Single Precision FP*

Core	ALUT	DSP (18x18)	Latency
Addition	0.25%	0	10
Multiplier	0.13%	0.5%	8
Divider	0.18%	1.7%	15
Inverse Root	0.16%	1.4%	19
EXP	0.26%	1.4%	16
LOG	0.59%	0.9%	21

Fixed-Point Multiplication

Precision	ALUT	DSP (18 x 18)	Latency
8	0%	0.13%	3
16	0%	0.26%	3
32	0%	0.52%	3
64	0.12%	4.17%	4

Double Precision FP*

Core	ALUT	DSP (18x18)	Latency
Addition	0.49%	0	10-14
Multiplier	0.29%	1.3%	7
Divider	0.44%	3.5%	21
Inverse Root	0.44%	3.5%	32
EXP	0.81%	3.6%	22
LOG	1.01%	1.6%	26

Resource utilization numbers are referred to Altera Stratix EP3SE260

*RSSI 2008, Martin Langhammer

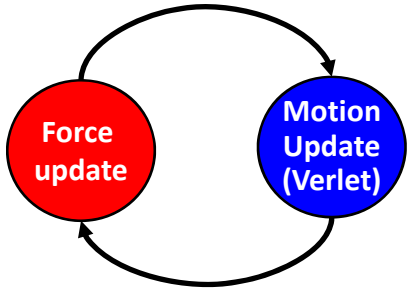
Enabling Factors for High Utilization

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:



Generally O(n), done on host

Initially O(n²), done on coprocessor

Many forces typically computed,

$$F^{total} = F^{bond} + F^{angle} + F^{torsion} + F^{H} + F^{non-bonded}$$

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

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

$$F_i^{C} = q_i \sum_{j \neq i} \left(\frac{q_i}{|r_{ji}|^3} \right) \vec{r}_{ji}$$

*IET/CDT06, ParC08, FPL09

Examine Short-Range Force Kernel

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

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

LJ force

short range component of Coulomb force

smoothing function

Force Pipeline

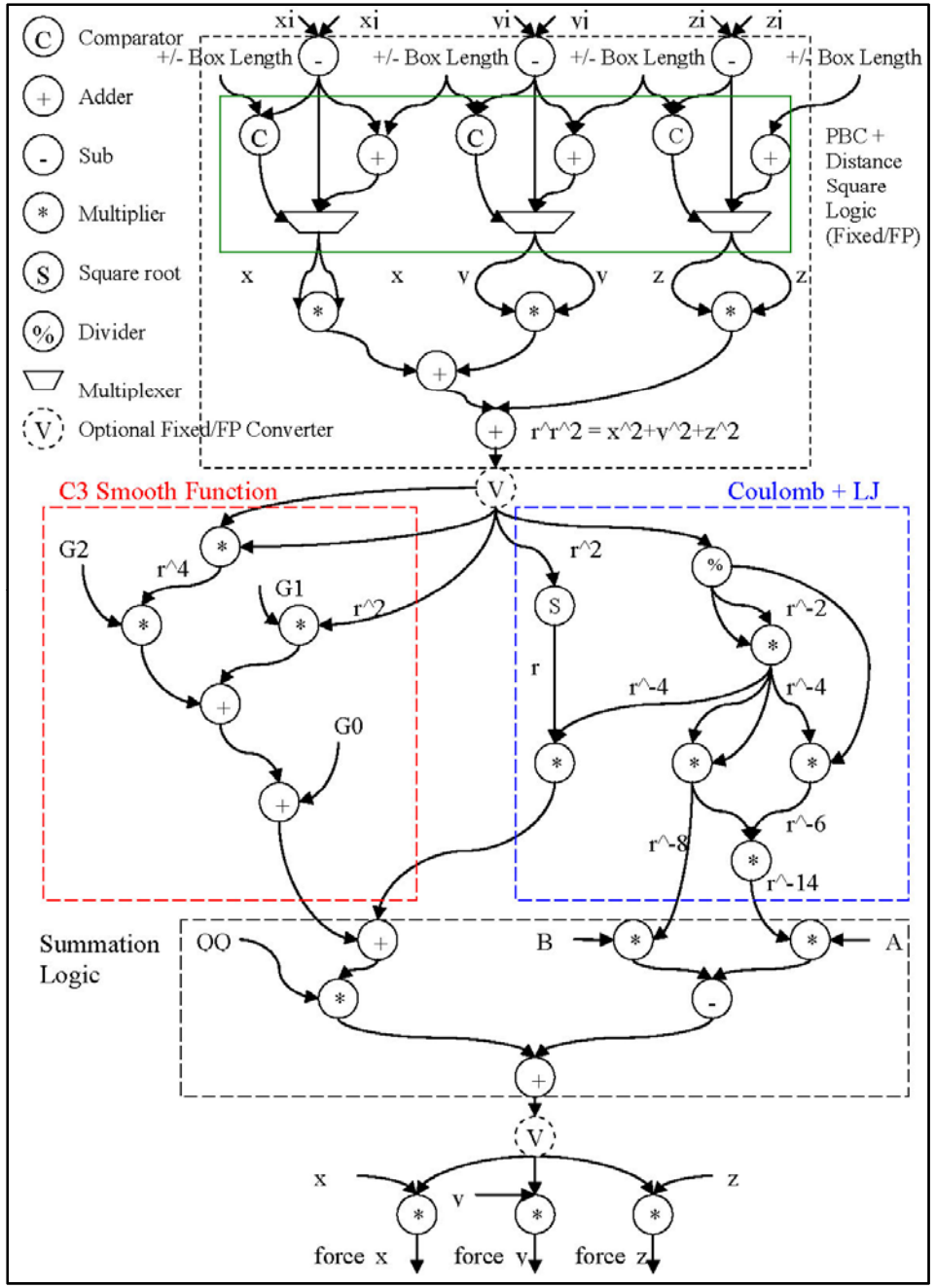
1. Choice of FP units
   Exactly those that you need:
   16 FADD, 16FMUL, 1 FDIV, 1 FSQRT

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

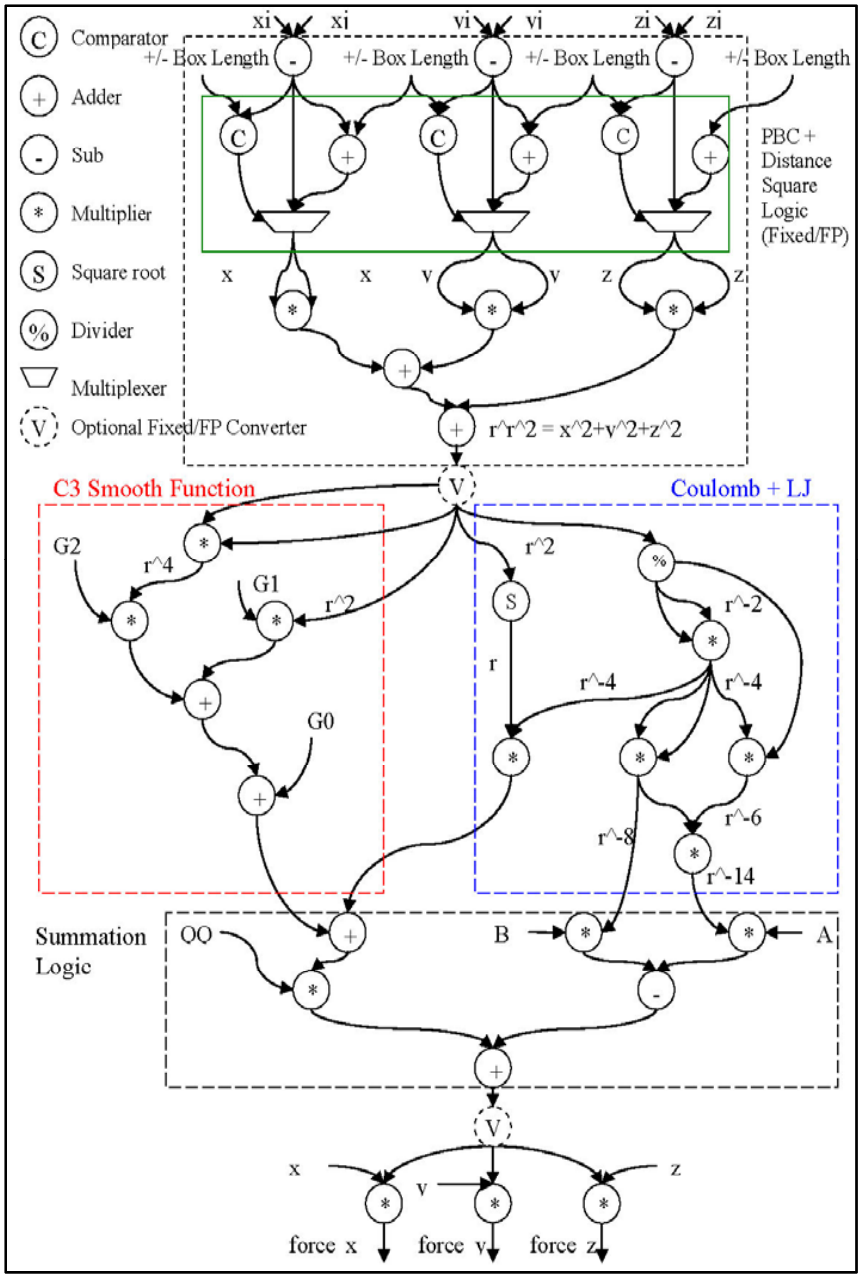
Double Precision

Resource Utilization Stratix-III 340

Mode	Precision	Logic Utilization (ALUT / Register)	Multipliers	Frequency (MHz)
IP Core	double	21% (10% / 18%)	27%	181

Number of force pipelines = 3

Throughput = 540M forces/s



Force Pipeline 2: Single Precision

Single precision

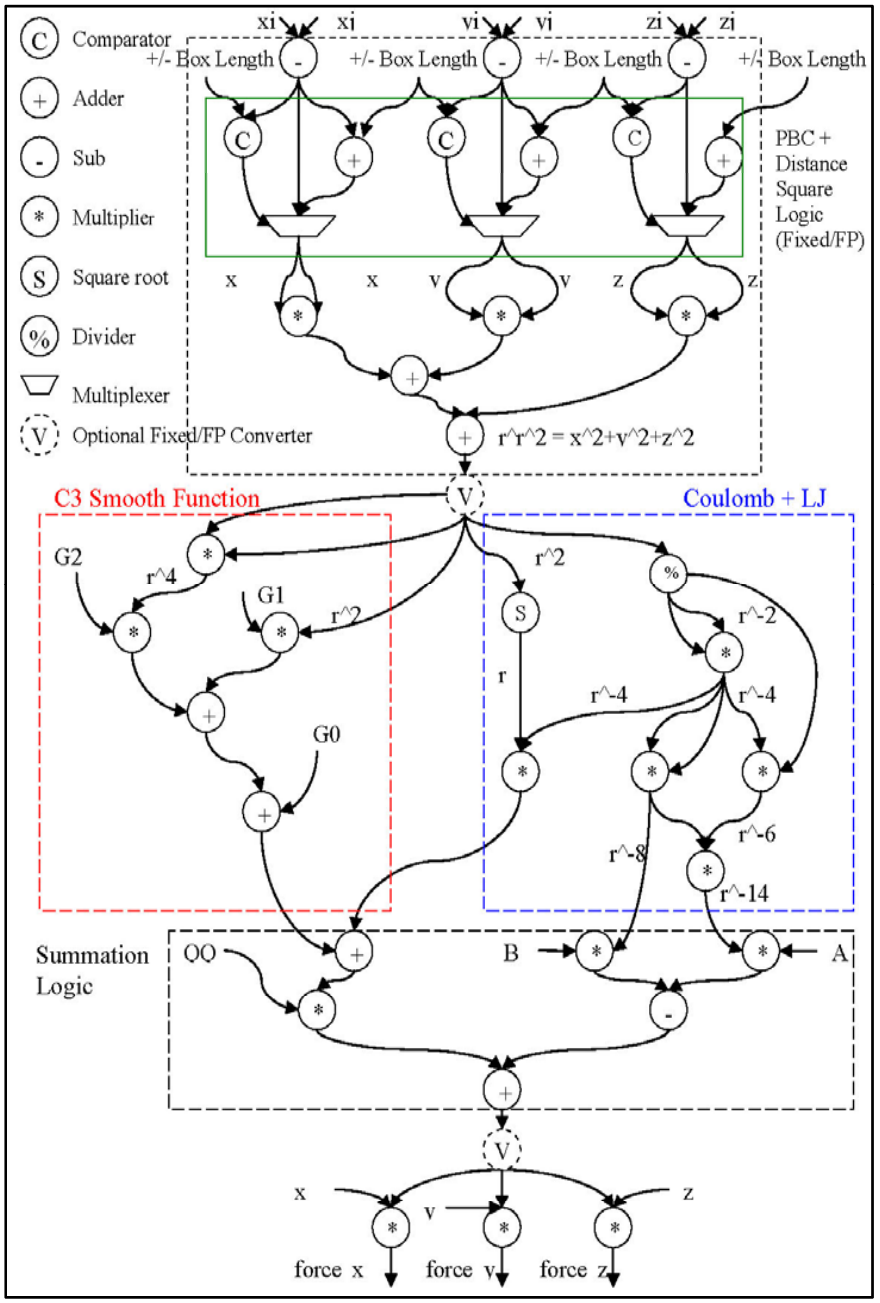
- Exclusion (small r) computed on host
- Precision may not be adequate

Resource Utilization Stratix-III 340

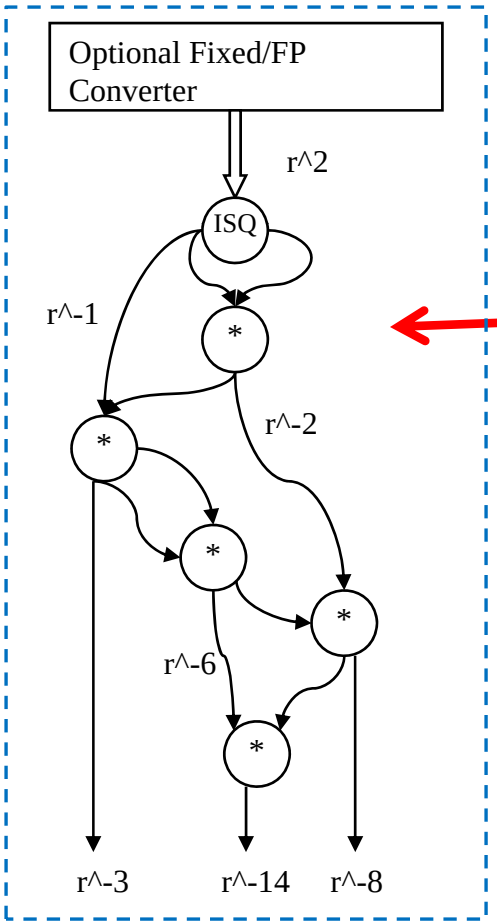
Mode	Precision	Logic Utilization (ALUT / Register)	Multipliers	Frequency (MHz)
IP Core	double	21% (10% / 18%)	27%	181
IP Core	single	10% (5% / 9%)	11%	184

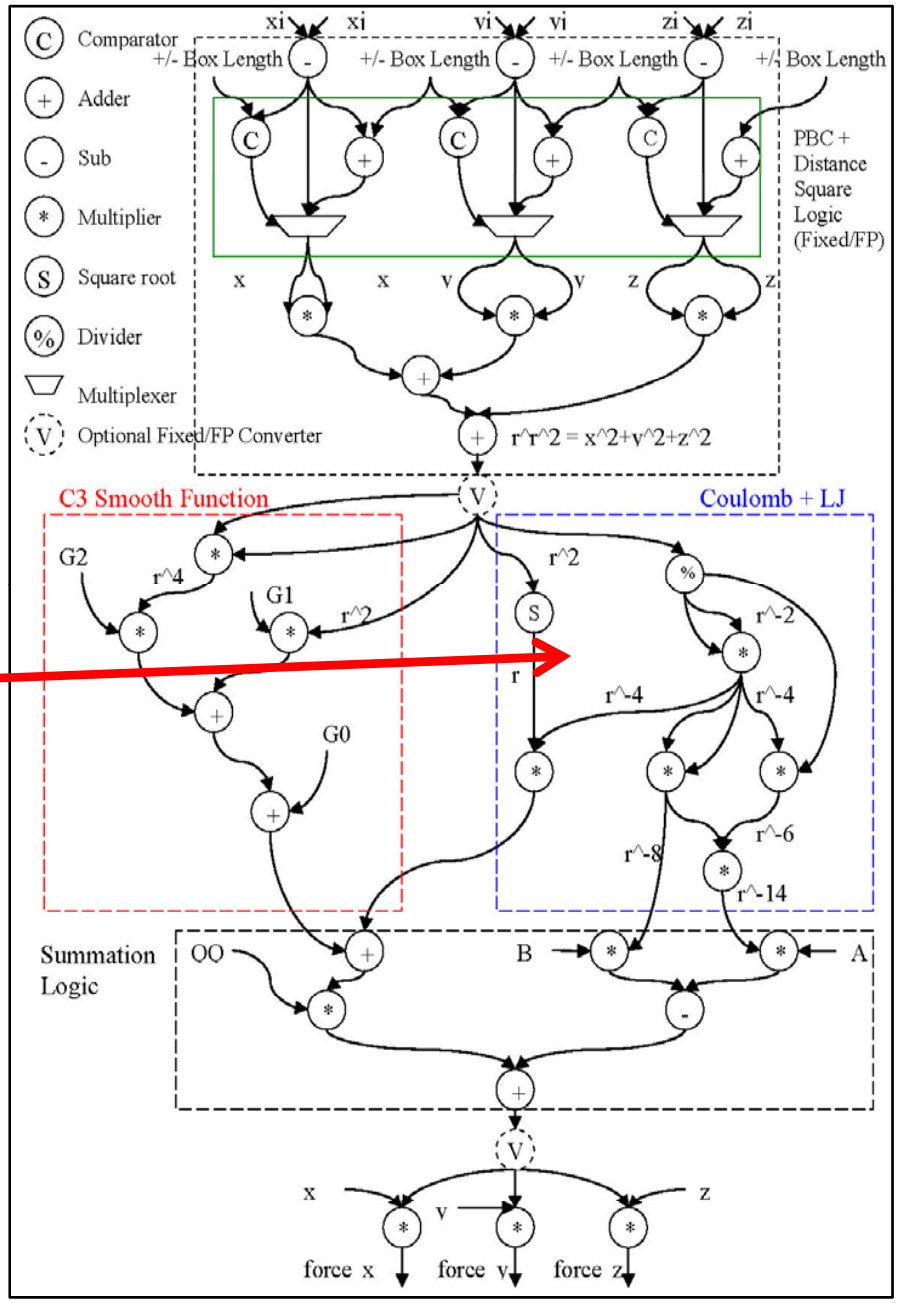
Number of force pipelines = 9

Throughput = 1.7G (3.15x DP)



Force Pipeline 3: Floating Point Compiler





BOSTON UNIVERSITY

Force Pipeline 3: Floating Point Compiler

Single precision + optimization w/ FPC

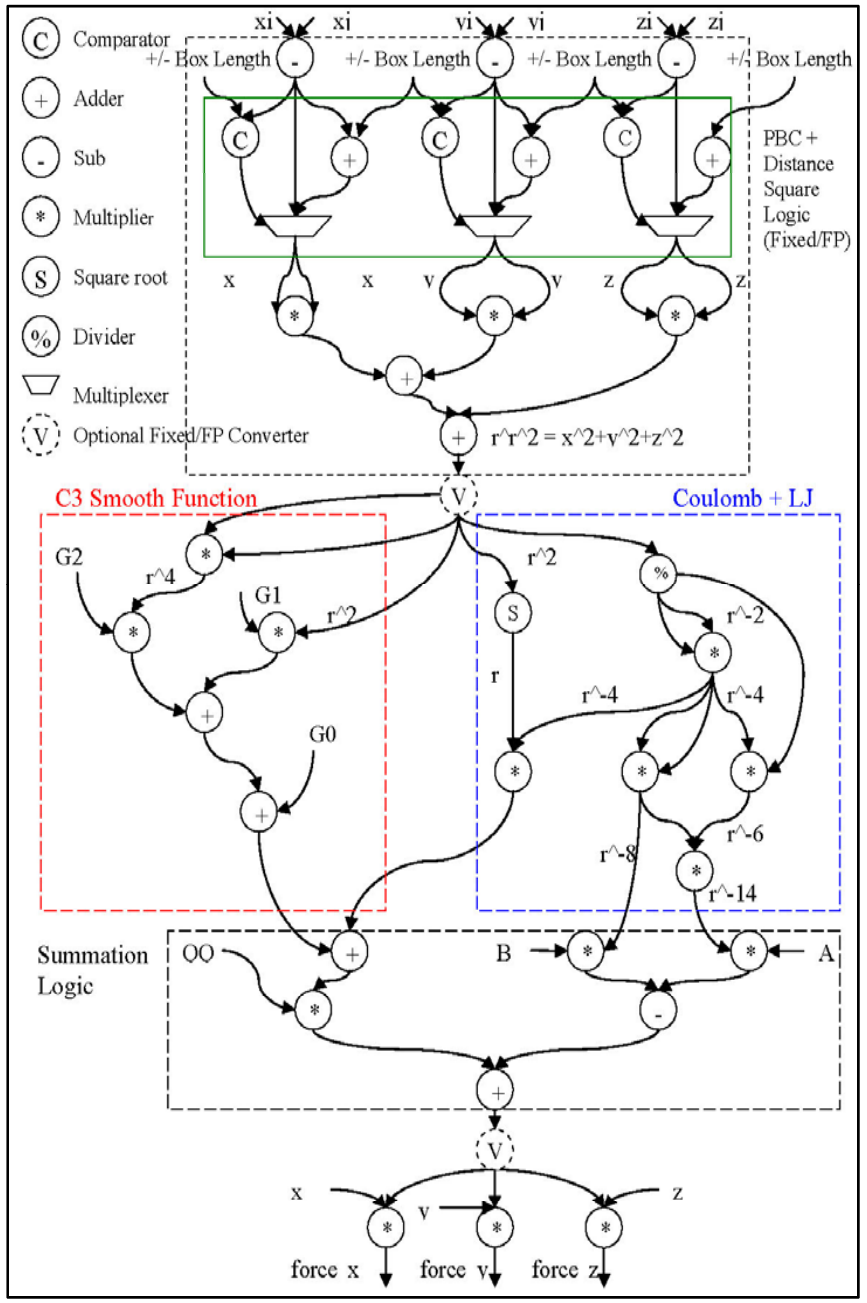
- Exclusion (small r) computed on host
- Floating point parts optimized with FPC

Resource Utilization Stratix-III 340

Mode	Precision	Logic Utilization (ALUT / Register)	Multipliers	Frequency (MHz)
IP Core	double	21% (10% / 18%)	27%	181
IP Core	single	10% (5% / 9%)	11%	184
FPC*	single	7% (4% / 6%)	11%	202

Number of force pipelines = 9

Throughput = 1.8G (3.3x DP)



Force Pipeline 4: Some Fixed Point

Hybrid: Single precision + 32 bit fixed point

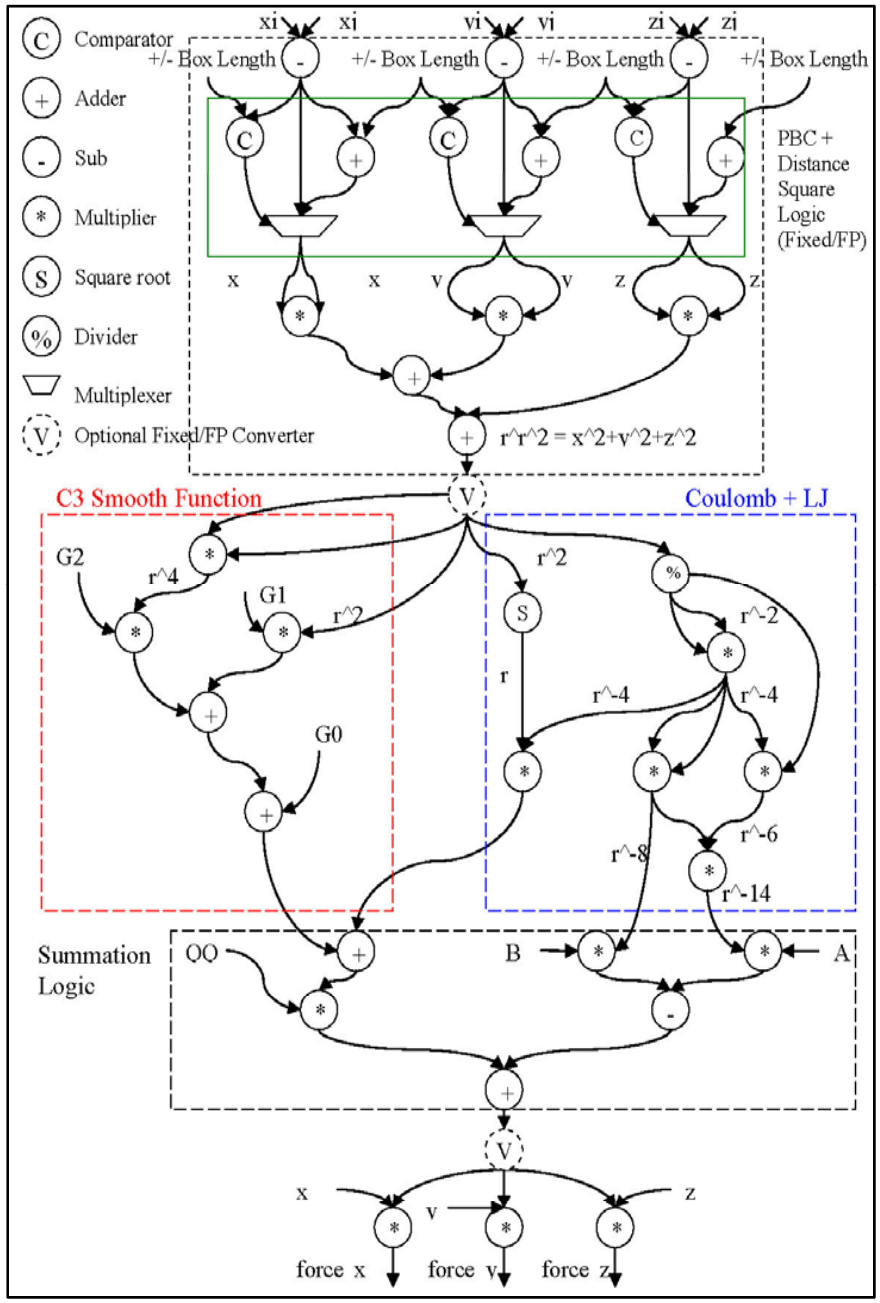
- Exclusion (small r) computed on host
- 32-bit precision likely to be adequate (see HPRCTA08)

Resource Utilization Stratix-III 340

Mode	Precision	Logic Utilization (ALUT / Register)	Multipliers	Frequency (MHz)
IP Core	double	21% (10% / 18%)	27%	181
IP Core	single	10% (5% / 9%)	11%	184
FPC*	single	7% (4% / 6%)	11%	202
Hybrid FPC*	32-bit integer /single	5% (4% / 5%)	9%	251

Number of force pipelines = 11

Throughput = 2.8G (5.2x DP)



BOSTON UNIVERSITY

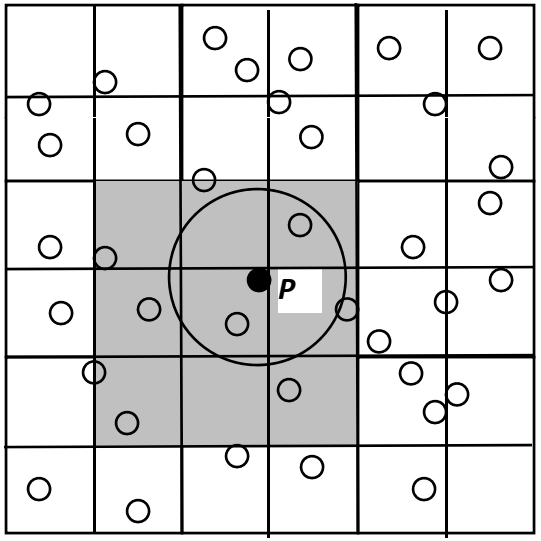
O(N) with Cell Lists

Observation:

- Typical volume to be simulated = 100Å^3
- Typical LJ cut-off radius = 10Å

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

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



Solution:

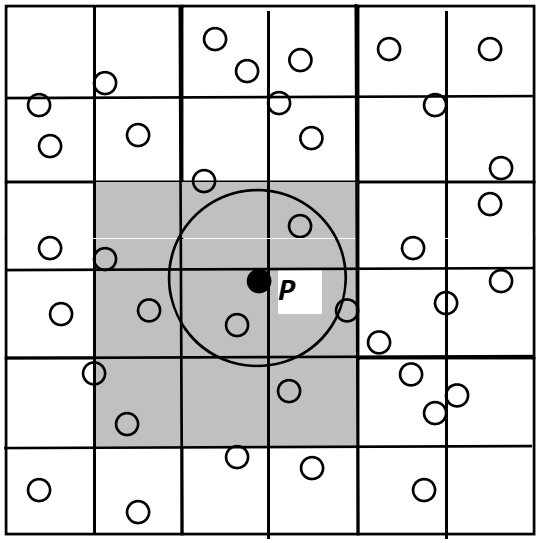
Partition space into "cells," each roughly the size of the cut-off

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

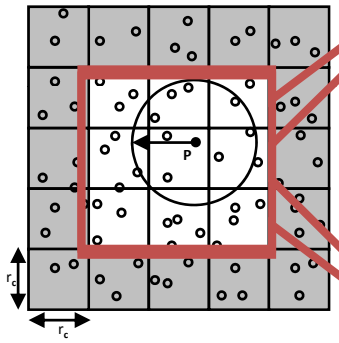
- Problem → Efficiency
  - ~15.5% efficiency when cell size is equal to cutoff
  - 84.5% of computation is wasted

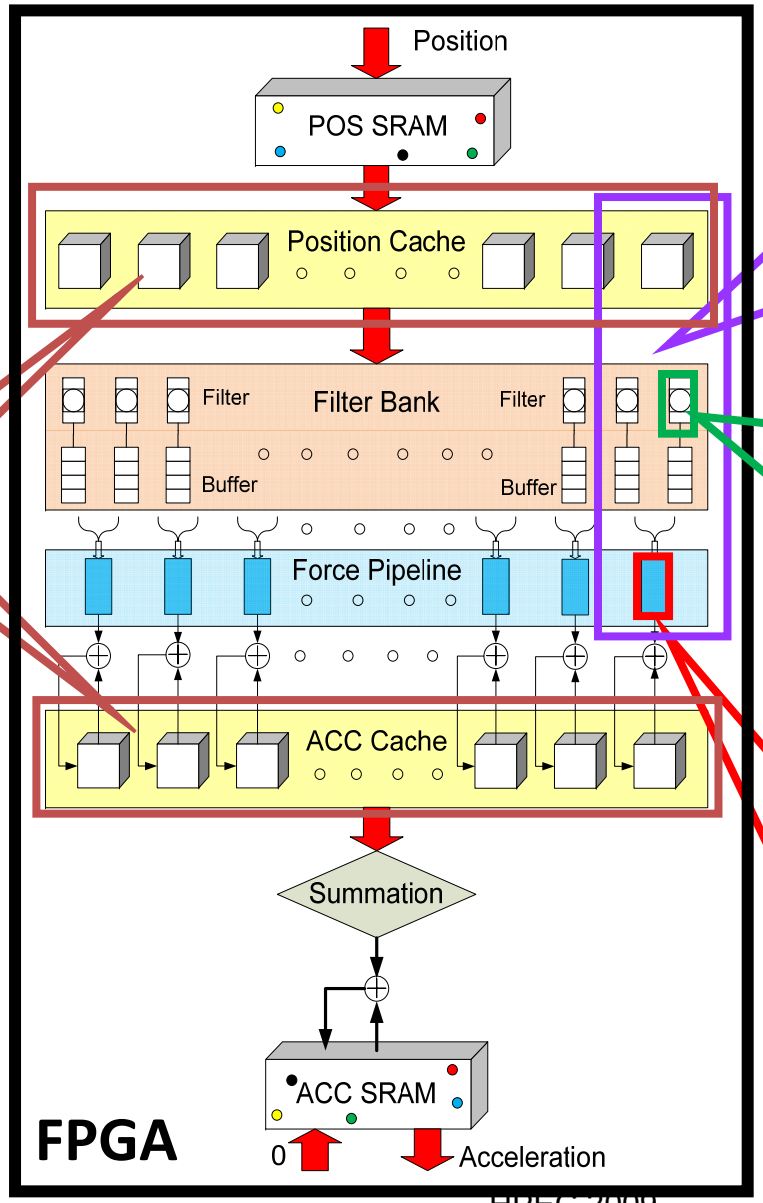


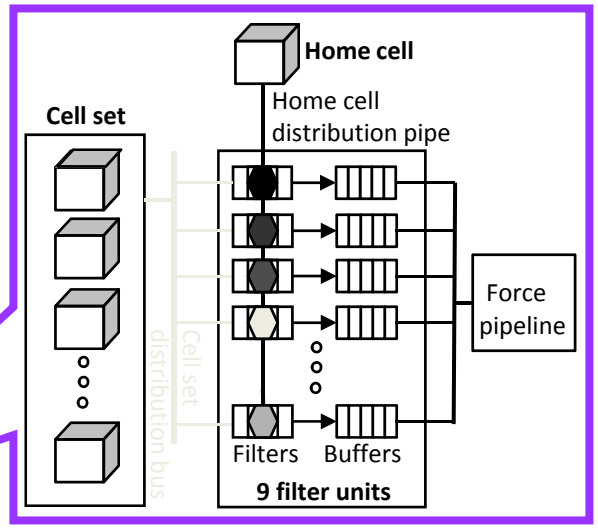
- Solution → Filtering*
  - Remove unnecessary particles outside cutoff
    - on-the-fly neighbor lists
  - Only compute forces for “valid” particle pairs

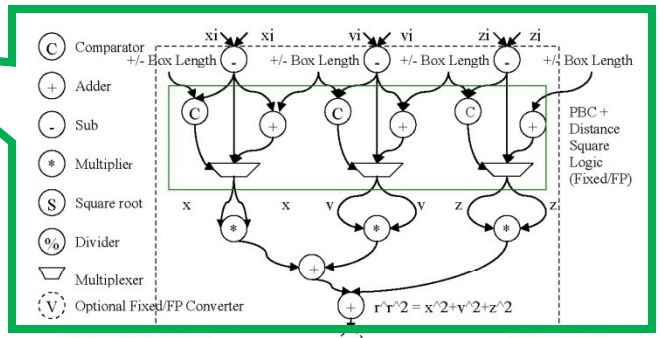
***FPL 2009**

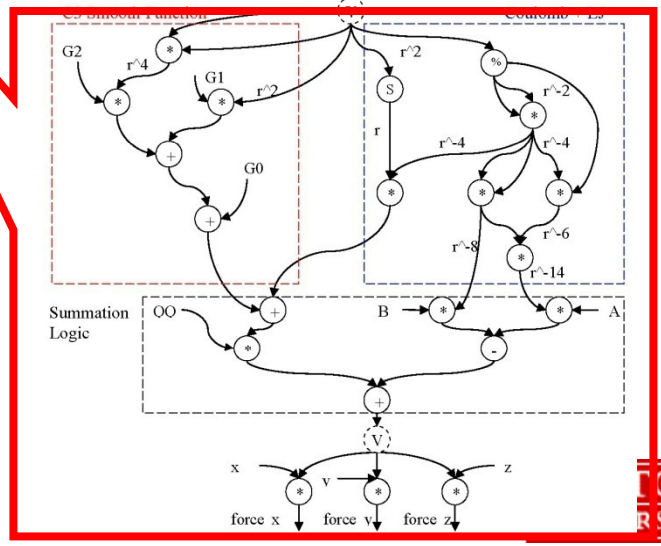
Idea: → 6x as many "filters" as force pipelines
→ Throughput improves 6x for marginal cost



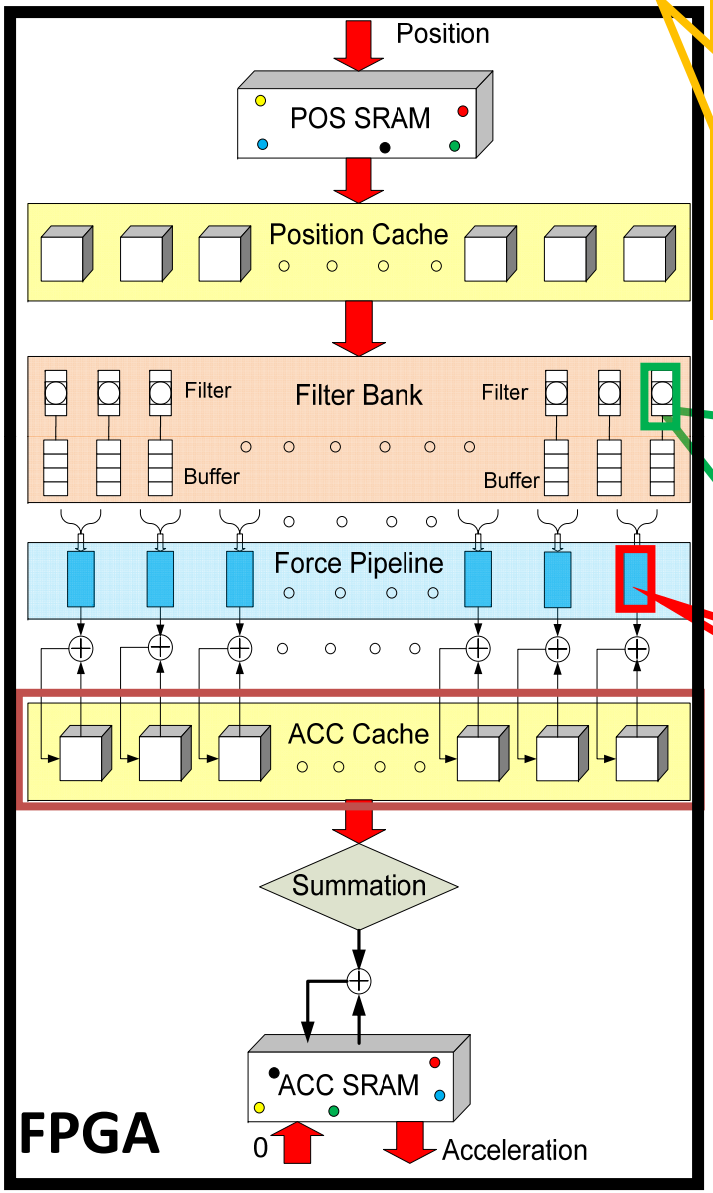


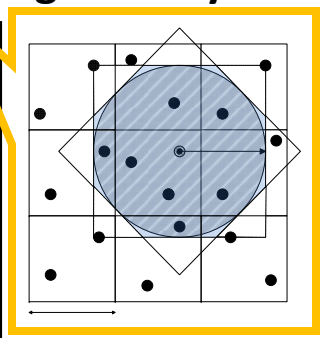


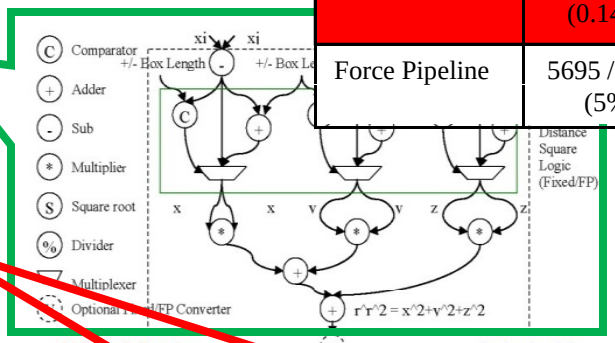


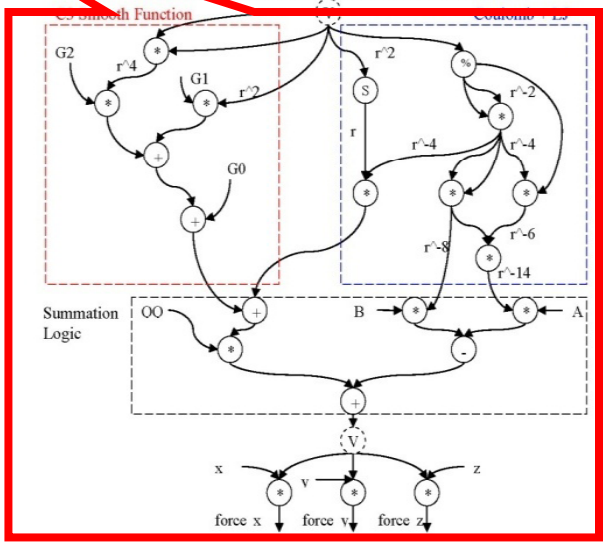


More ideas -- reduce precision of filter
-- approximate spherical geometry







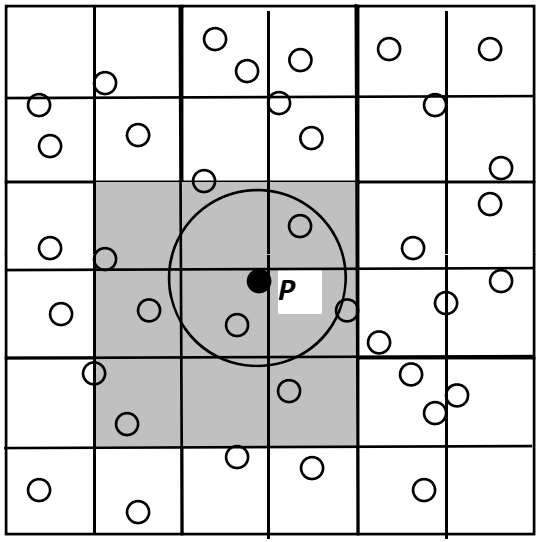


Method	ALUTs / Registers	Multipliers	Filter Efficiency	Extra Work
Full precision	341 / 881 (0.43%)	12 (1.6%)	100%	0%
Full precision - logic only muls	2577 / 2696 (1.3%)	0	100%	0%
Reduced precision	131 / 266 (0.13%)	3 (0.4%)	99.5%	3%
Reduced precision- logic only muls	303 / 436 (0.21%)	0	99.5%	3%
Planar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

MD Design (so far in this talk)

For all particle pairs (P_a,P_b)

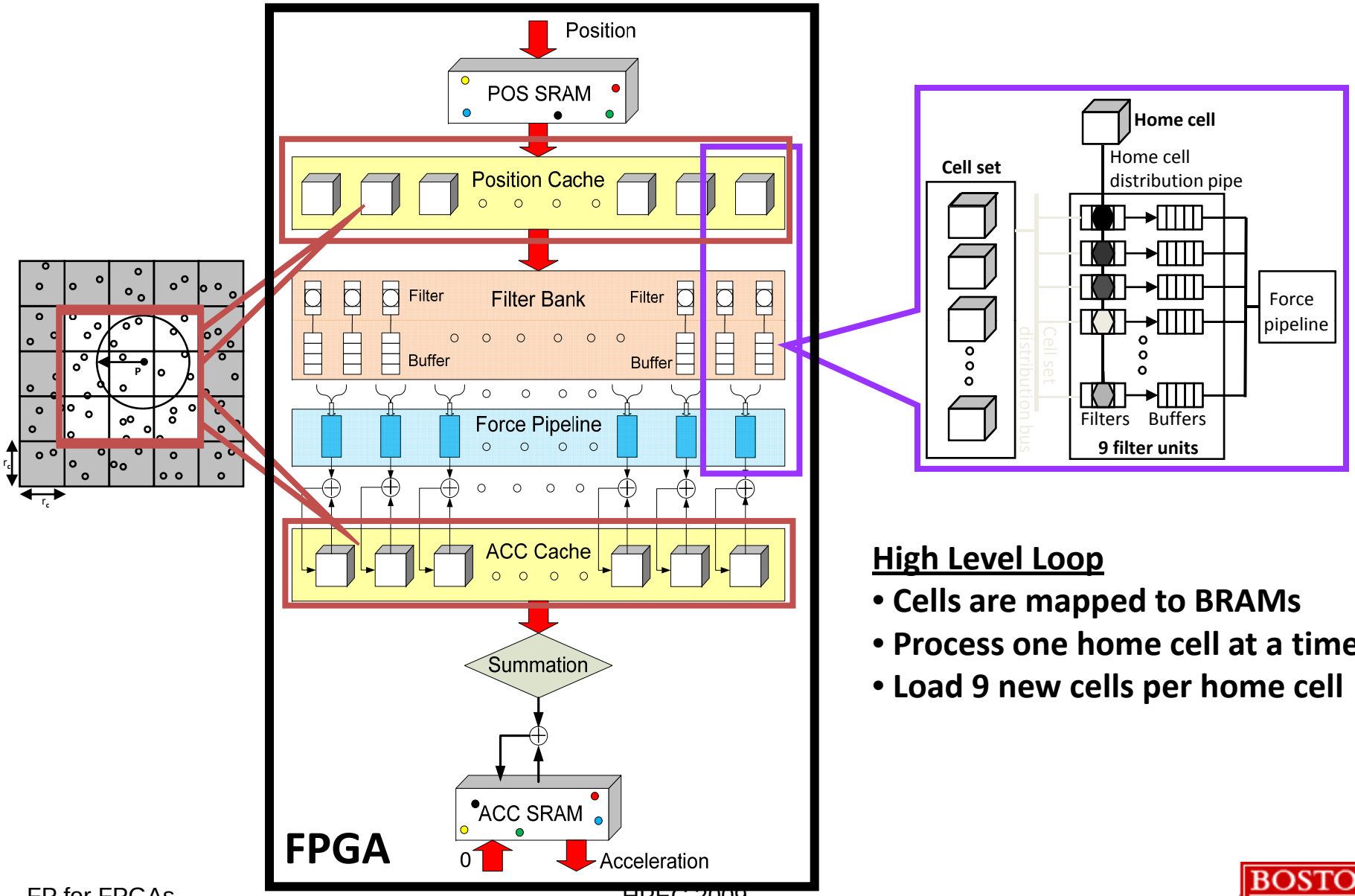
- Small r gets done on host in DP
- r < cut-off gets done with force pipeline
- (P_a,P_b) in cell-set gets done with filter
- (P_a,P_b) not in cell-set computed on host → O(1)



A high end FPGA (65nm) fits

- 8-10 force pipelines
- 80 filters

Much more to get this working …



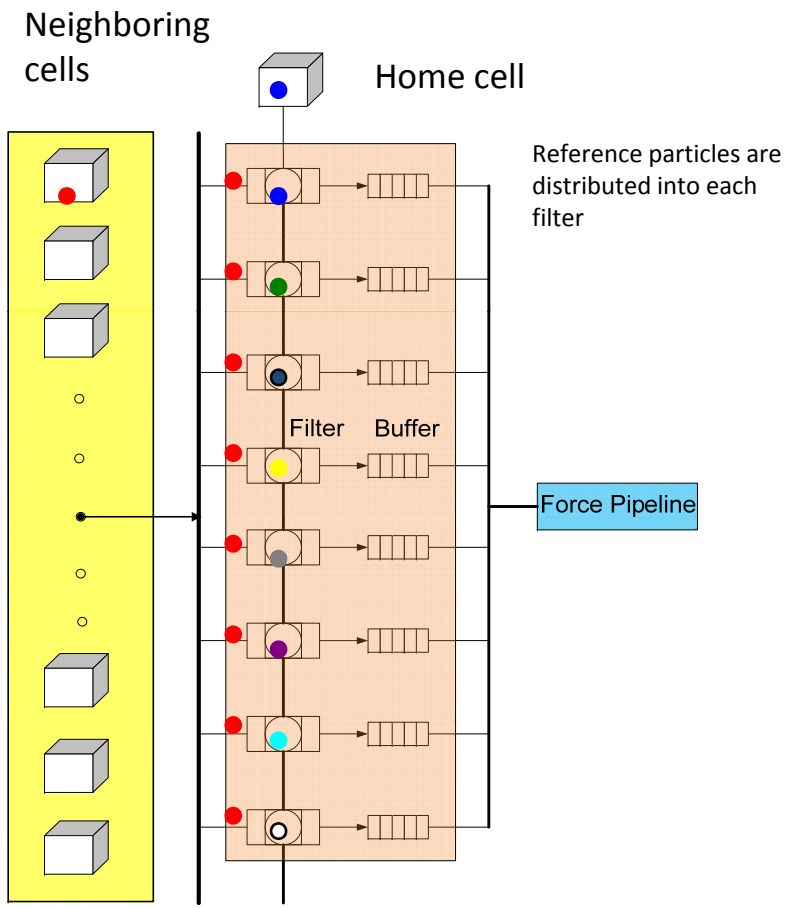
High Level Loop

- **Cells are mapped to BRAMs**
- **Process one home cell at a time**
- **Load 9 new cells per home cell**

BOSTON UNIVERSITY

Mapping Particles Pairs to Filters

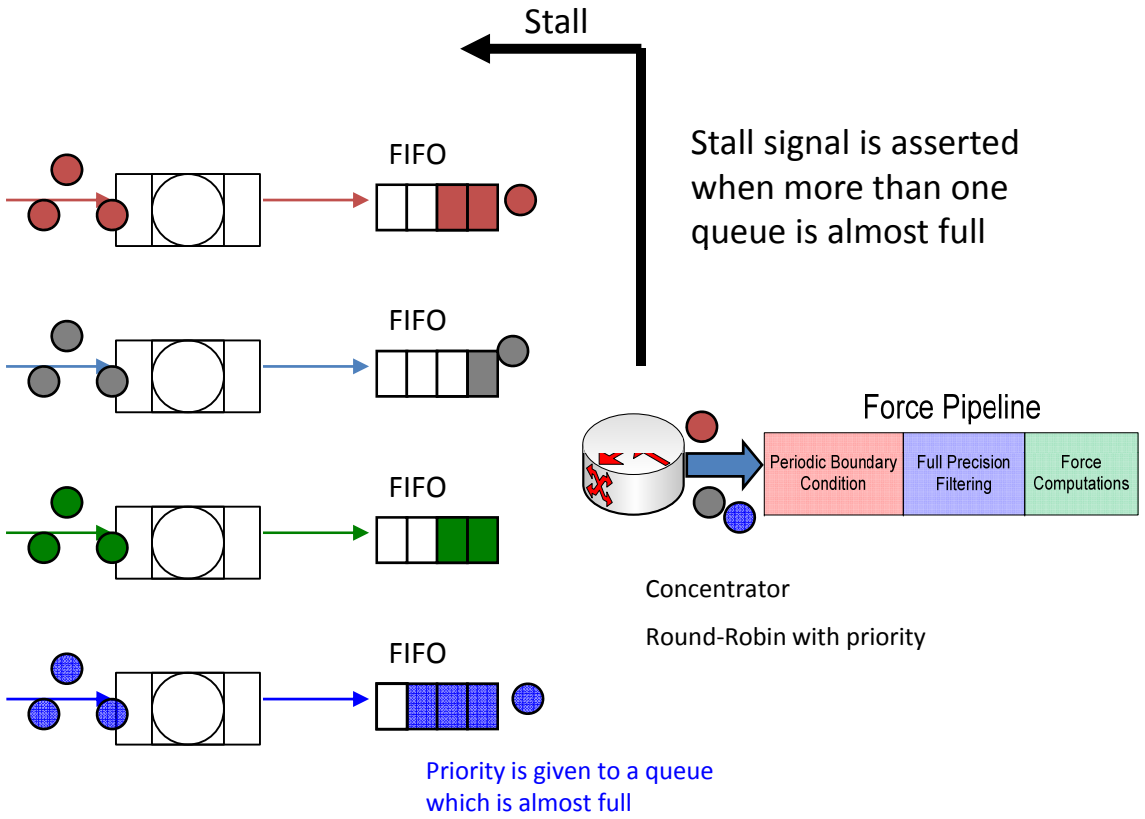
- Particle Mapping (PM)
  - Each filter is responsible for a different reference particle
  - Each cycle, a single partner particle from the cell set is broadcast to all of the filters
  - Particle pairs which satisfy filtering rules are stored into filter queues for force computations



8-9 Filters / force pipeline

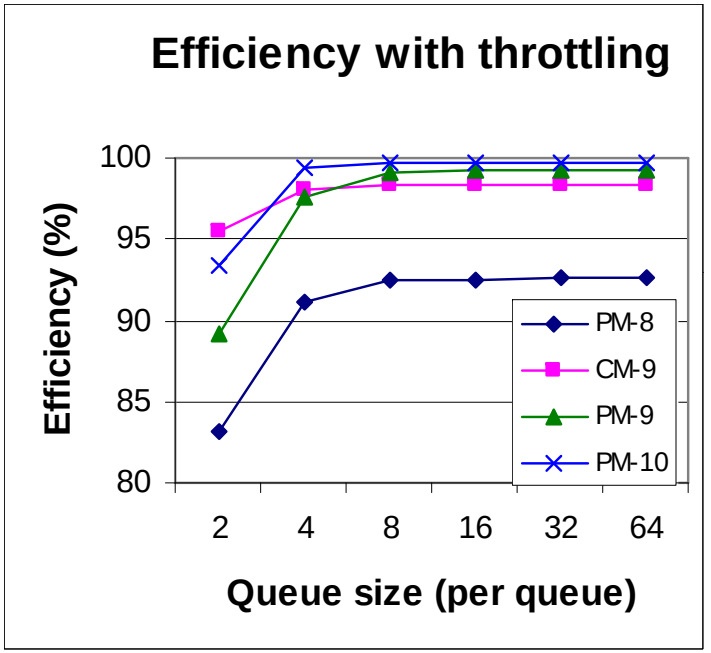
Filtering List – Continuous Queueing

- "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



Queue Size

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



MD Results

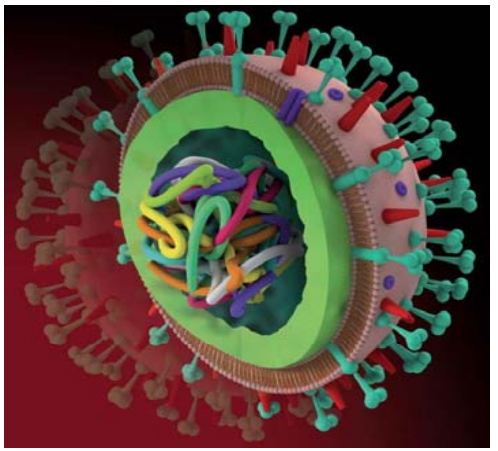
- 8 force pipelines fit easily into a Altera Stratix III EP3SE260
  - Place & Route
  - Simulation quality validated with standard methods (see HPRCTA08)
  - Runs @ ~200 MHz
  - Each filter bank contains 8-9 filters and one force pipelines
  - Over 95% utilization (data transfers discussed elsewhere)

- 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.

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

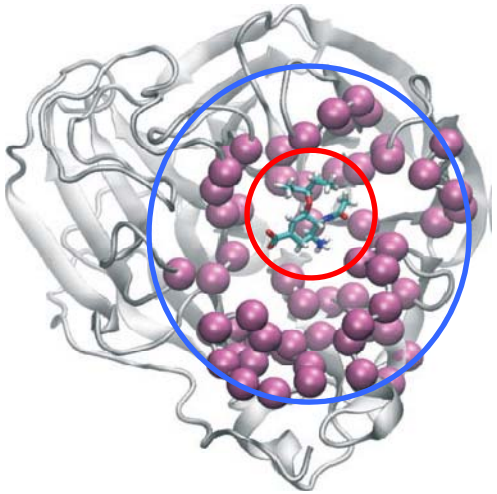
Why is Docking so important?+


#

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
*Landon, et al. Chem. Biol. Drug Des 2008
#From ***New Scientist*** www.newscientist.com/channel/health/bird-flu

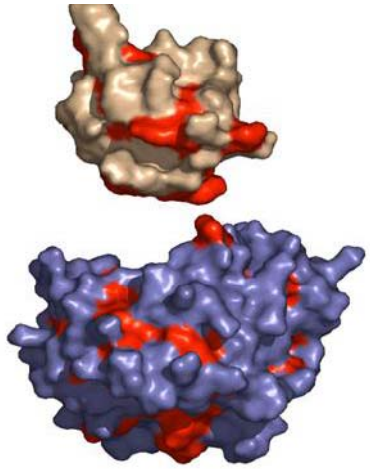
Modeling Rigid Docking

Rigid-body approximation

Grid based computing

Exhaustive 6D search

Low Precision Data



Pose score = 3D correlation sum

$$E(\alpha,\beta,\gamma)=\sum_{p}\sum_{i,j,k}R_P(i,j,k)\cdot L_P(i+\alpha,j+\beta,k+\gamma)$$

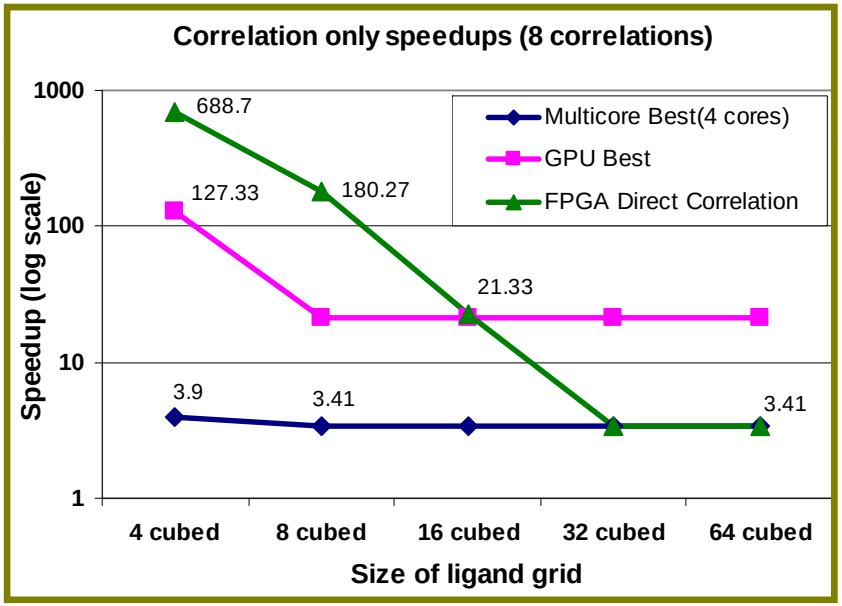
FFT to speedup the correlation

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

Image courtesy of Structural Bioinformatics Lab, BU

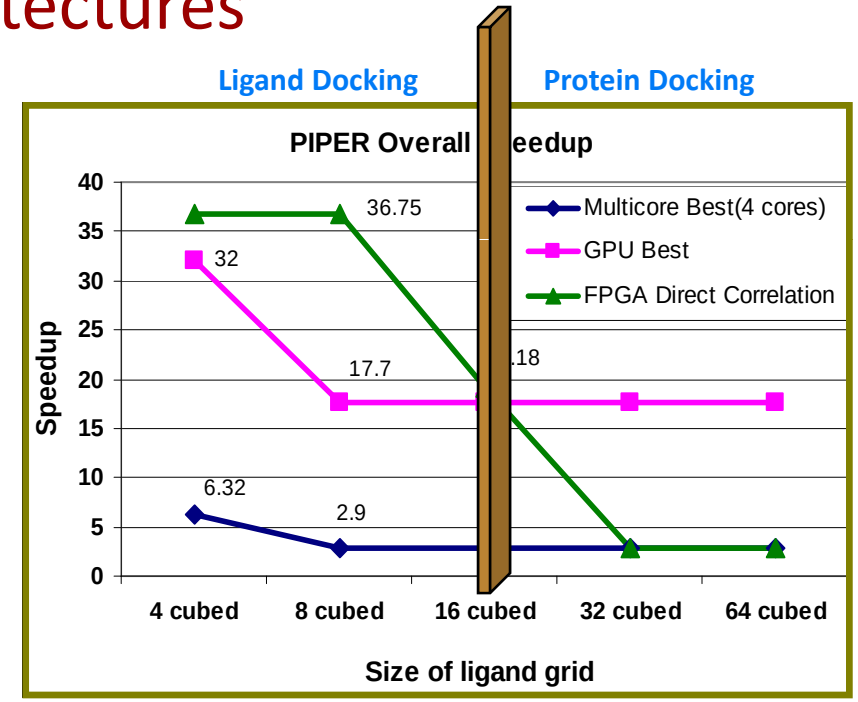
Results

Speedup on different architectures



* Baseline: Best Correlation on single core

Ligand Docking | Protein Docking



* Baseline: PIPER running on single core

Discrete Event Simulation of MD*

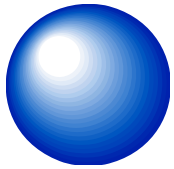
- Simulation with simplified models
- Approximate forces with barriers and square wells
- Classic discrete event simulation

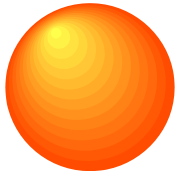
*FPL07, FCCM08

An Alternative ...

Only update particle state when "something happens"

- **"Something happens" = a discrete event**



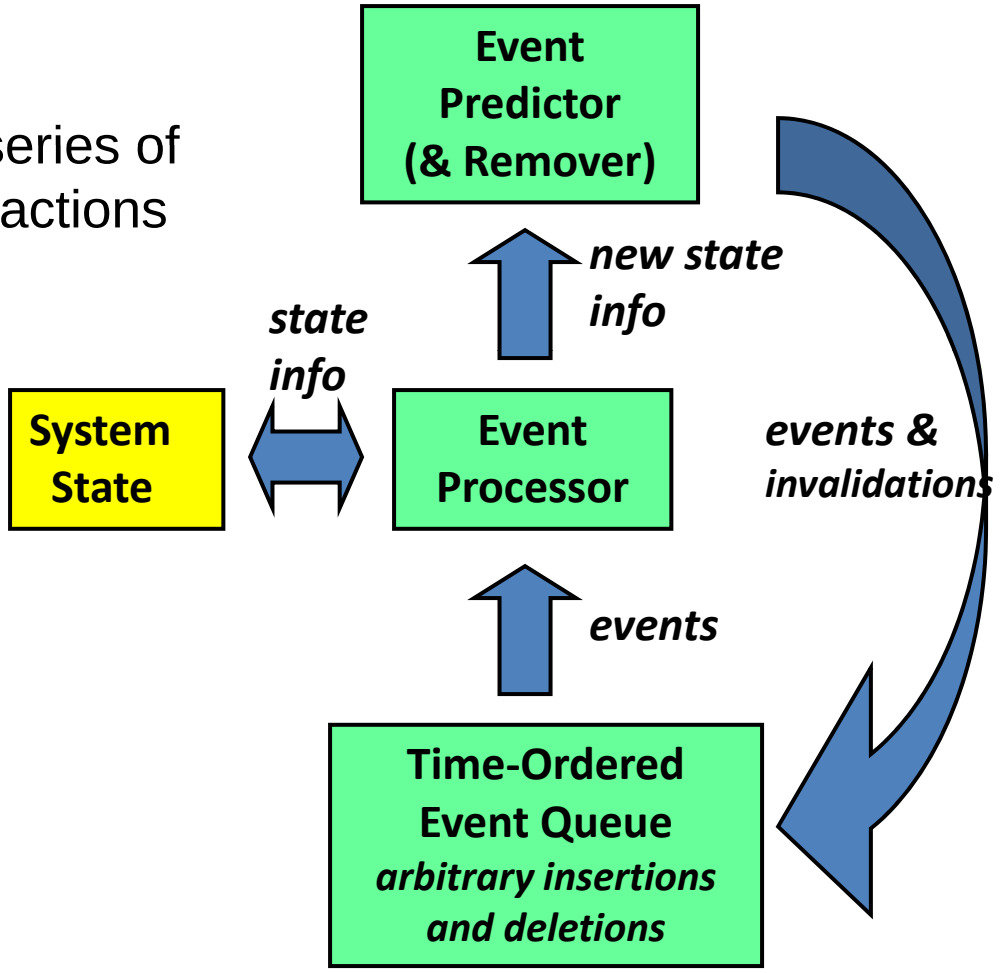


- **Advantage → DMD runs 10^5 to 10^9 times faster than tradition MD**

- **Disadvantage → Laws of physics are continuous**

Discrete Event Simulation

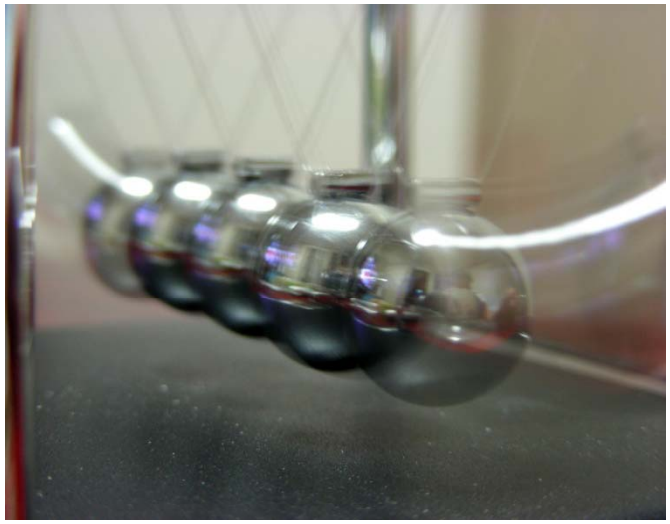
- Simulation proceeds as a series of discrete element-wise interactions
  - **NOT** time-step driven

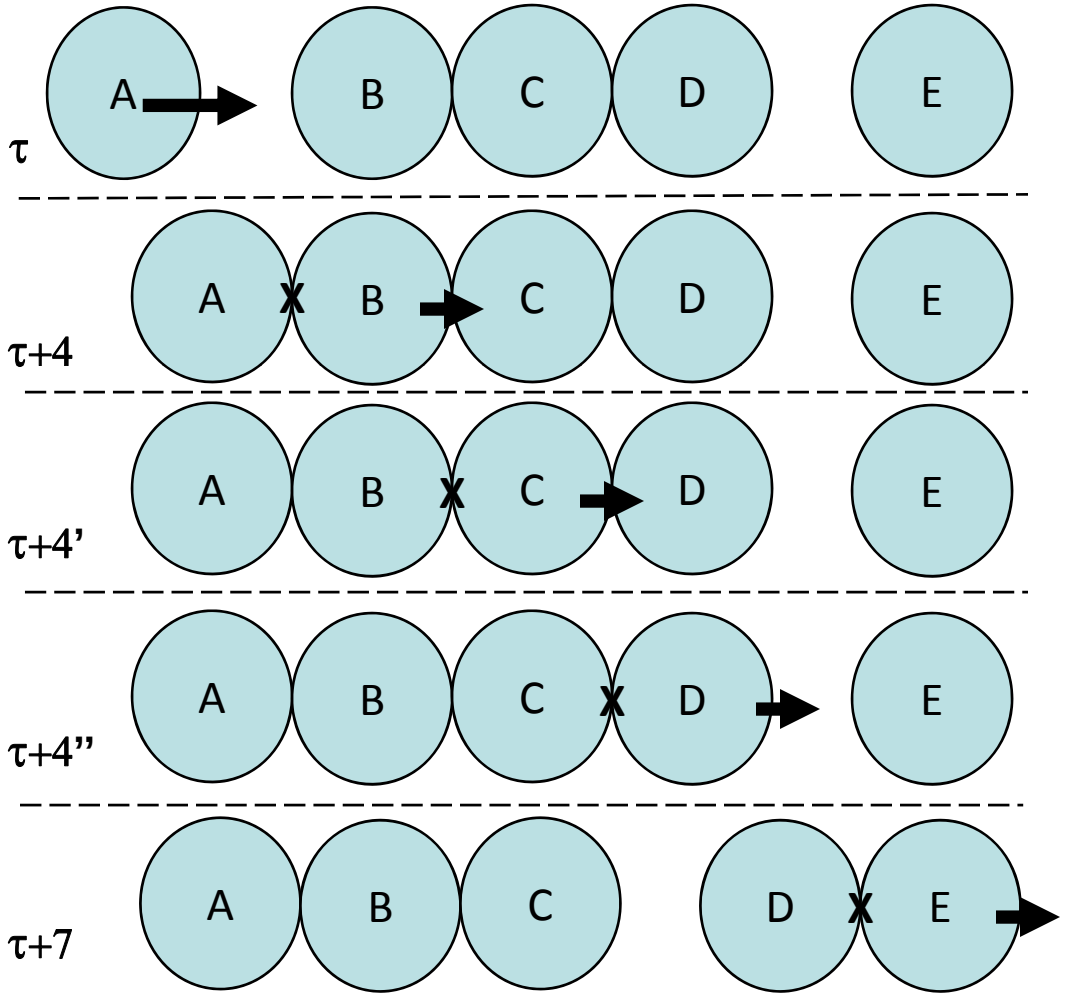


- Seen in simulations of …
  - Circuits
  - Networks
  - Traffic
  - Systems Biology
  - Combat

PDES - Why is it hard for DMD?

Event propagation can be infinitely fast over any distance!



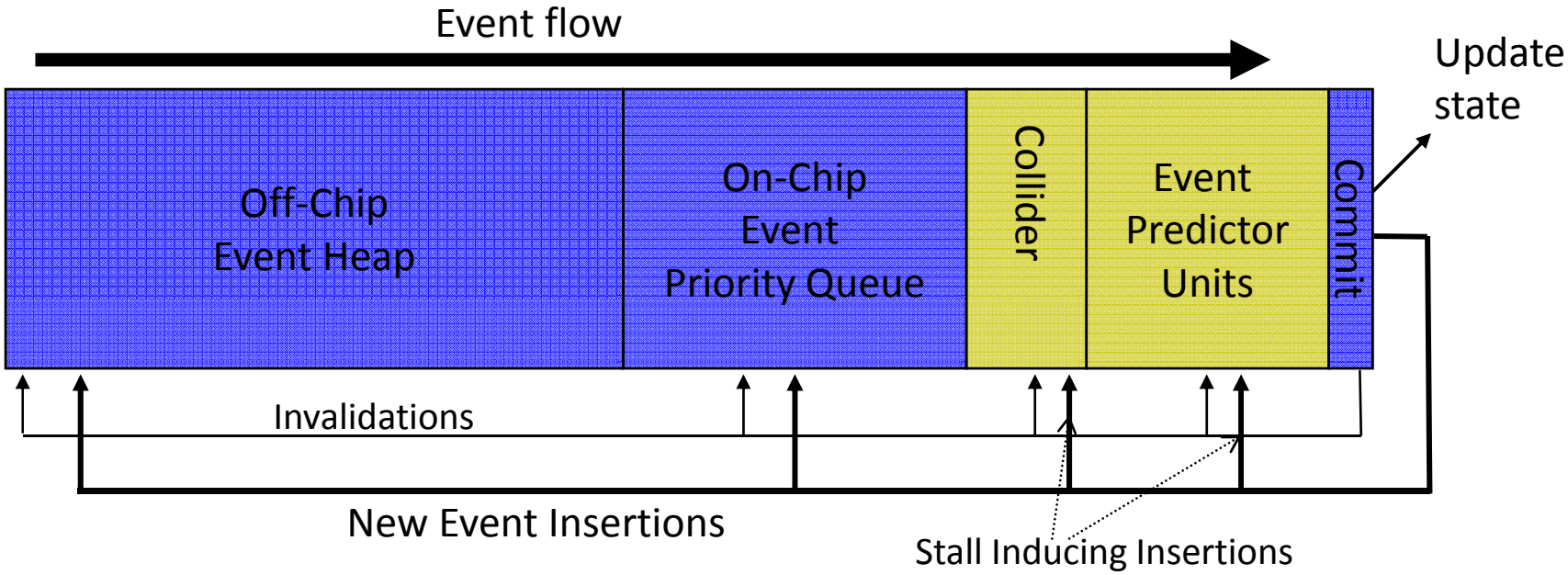


Note: a "chain" with rigid links is analogous and much more likely to occur in practice

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

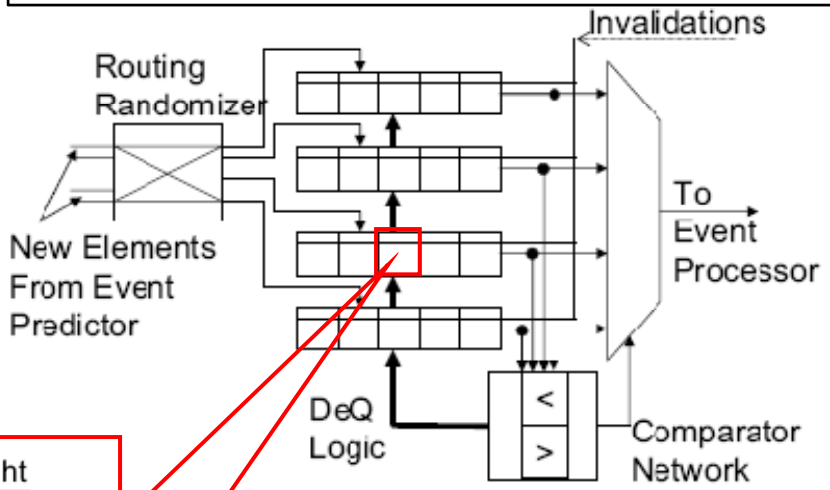


On-Chip, "Scrunching" Priority Queue

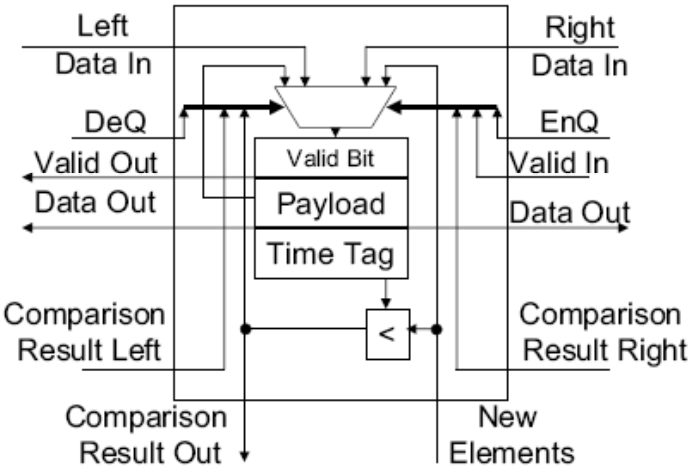
Queue operation → Each cycle:

- **Dequeue next event**
- **Up to four insertions**
- **Unbounded invalidates**
- **Fill holes with "scrunching" (conditional two-step advance)**

Insertion/deletion priority queue



Queue element



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?