Bulldozer Overview

Ted Barragy

January 23, 2012

Overview

Bulldozer Architecture Overview

- Block Diagrams
  - 2S Compute Node to Bulldozer Module
- Architecture Details
  - Cache, TLB , FPU & Miscellaneous Details

Bulldozer SW Optimization Notes

- A Short List
- Additional Hazards & AVX-256
- Additional Resources

BD Architecture Overview

Block Diagrams

Typical AMD Interlagos Compute Node

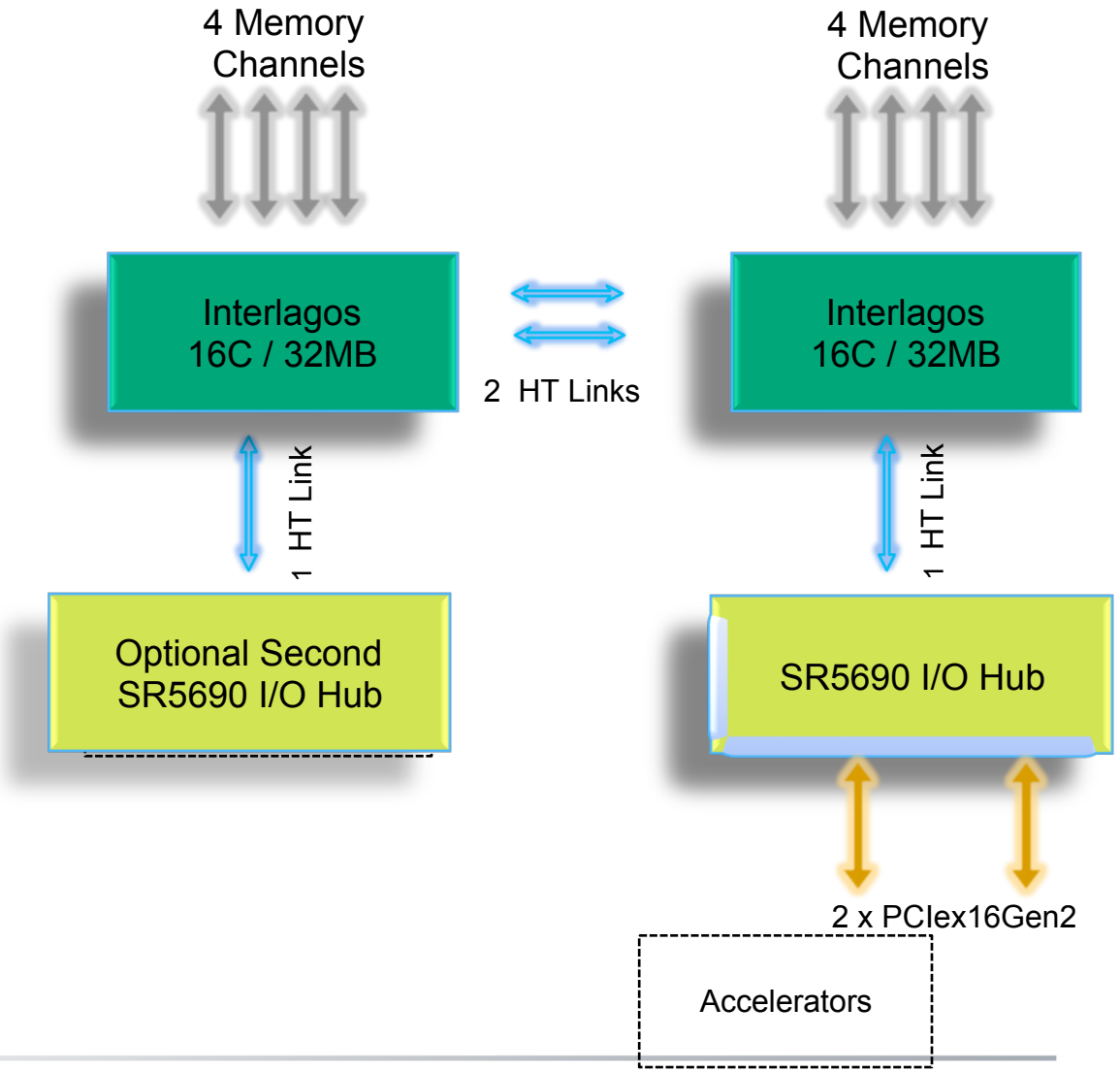
Two 'Interlagos' CPUs

Each Interlagos CPU Has

- 16 x Next Gen Bulldozer Cores
  - AVX, FMA4
  - 2.3 GHz base frequency
- 32MB combined L2 + L3 cache
  - Non-inclusive hierarchy
- 4 x Memory Channels
  - DDR3-1600, 2 DPC
- 4 x HT3 Links
  - 6.4 GT/s, 16 bits
  - 12 GB/Sec, per link, per direction

2xSocket Compute Node [1,2]

- HPL – 239 GFLOP / Sec
- Stream – 73 GBytes / Sec



Detailed 2 Socket HT3 Interconnect

Multichip Module

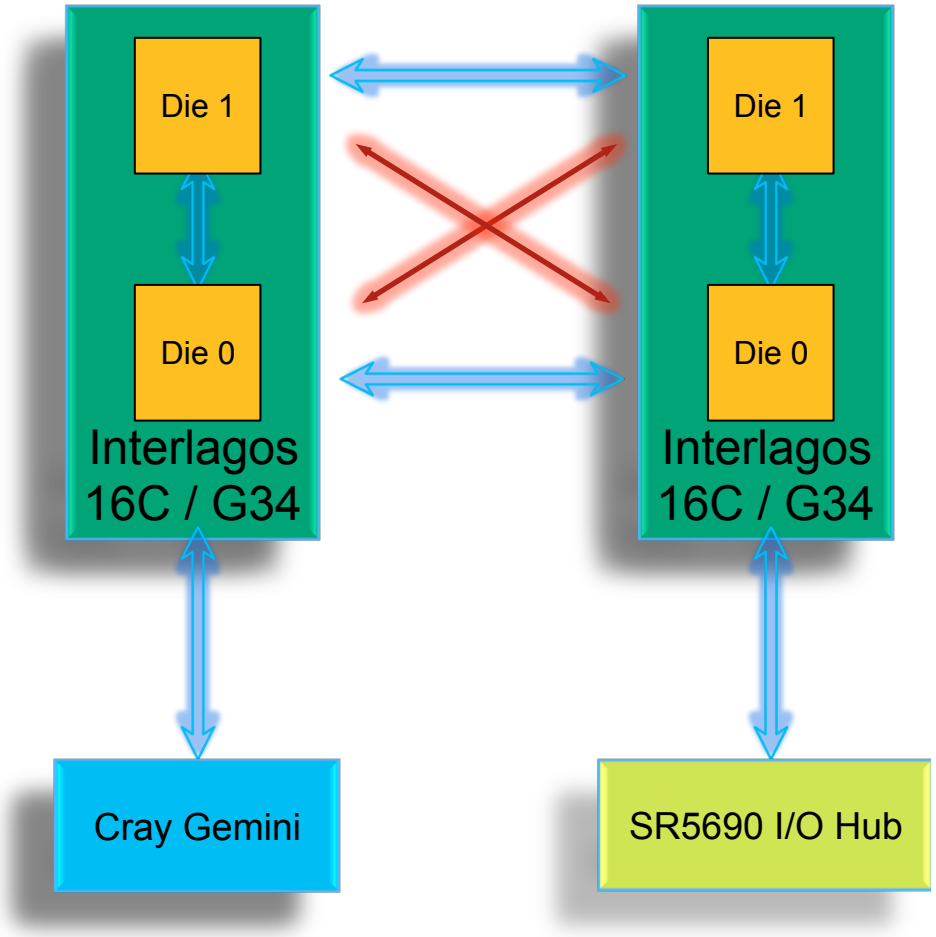
Each Interlagos CPU has 2 silicon die
- One master, one slave
- Appears as 16C CPU to the OS
- Leverages a single design effort
- Preserves 4 x HT link system arch

2 x HT Links Between CPUs

Implemented as a 'box' + 'diagonals'
The diagonal links are half width
- Some things are done to improve this

NUMA Aware Programming

Can extend inside each socket



16 bit HT Link 8 bit HT Link

AMD
The future is fusion

Titan Compute Node: Cray Xk6 (1 x G34 Socket)

1xG34 Socket

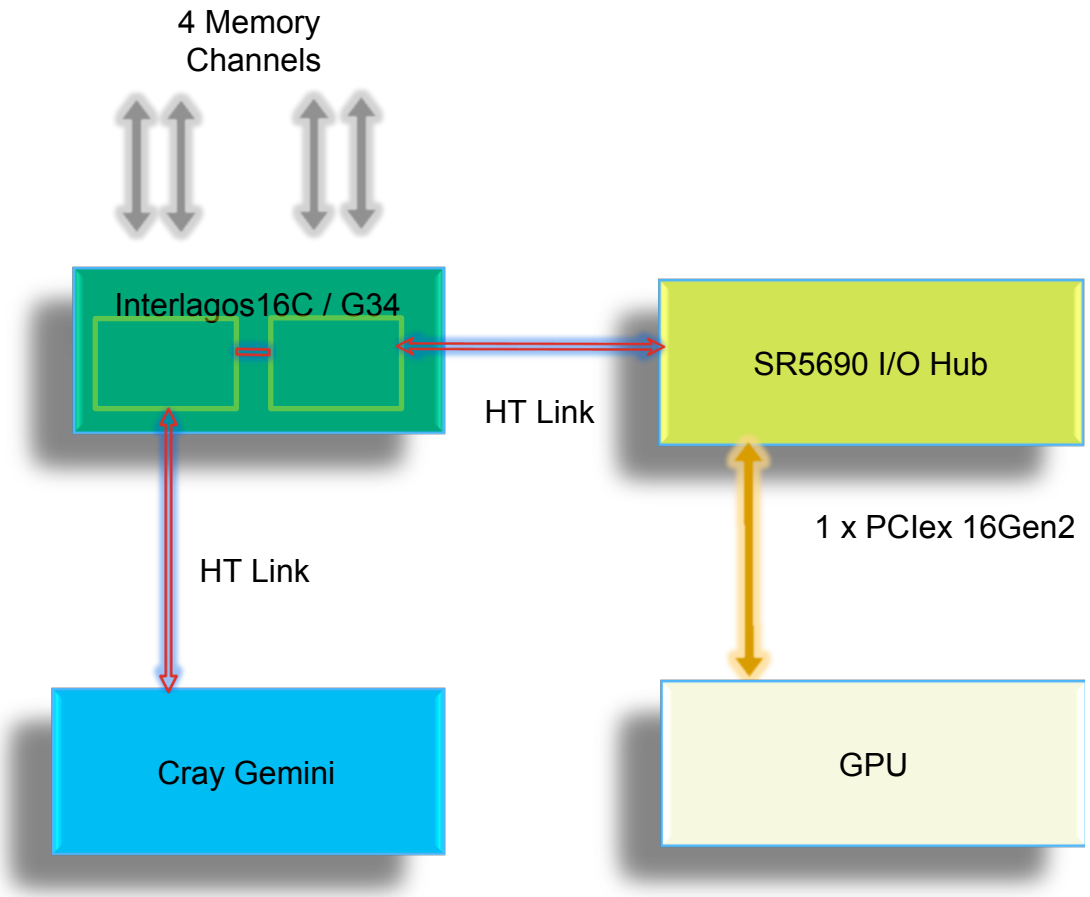
16 x Next Gen Bulldozer Cores
16MB combined L2 + L3 cache
4 x Memory Channels

1 x GPU

Nvidia Fermi / Kepler

Dual die hop count from Gemini to GPU may be important for maximum performance.

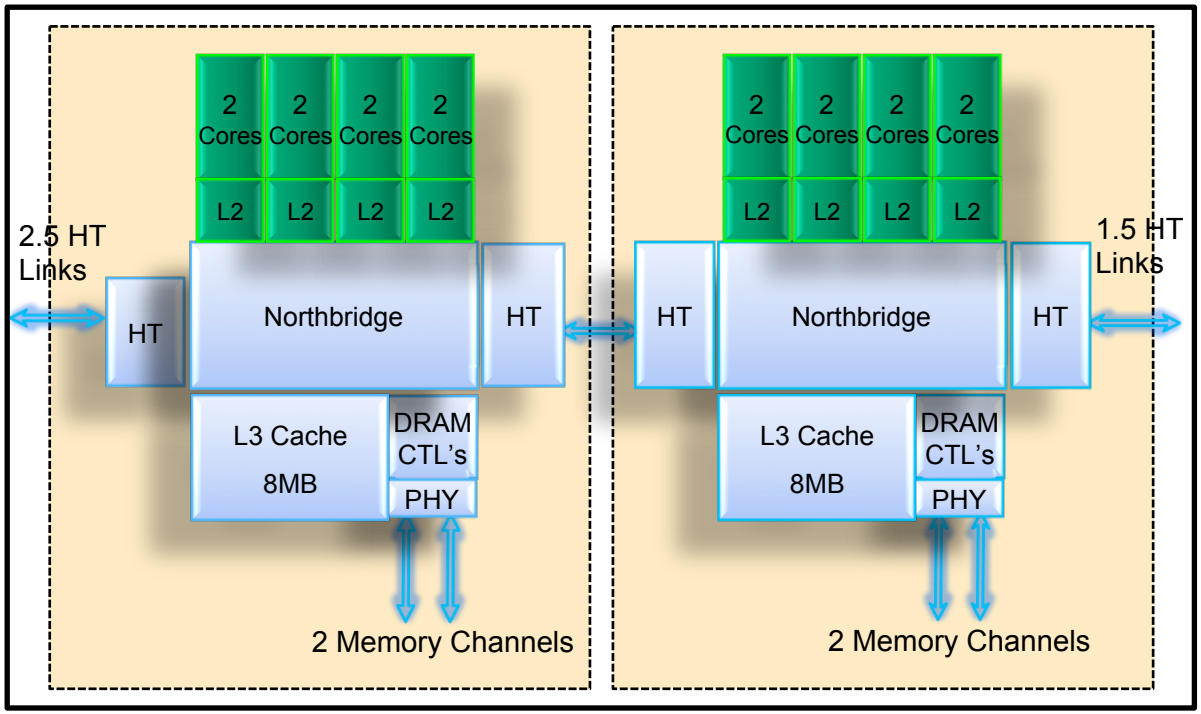
Work with your Cray performance tuning team



Interlagos CPU (Socket G34)

2 x Silicon Die Per Socket

- Multi-Chip Module
  - 32nm technology
- 8 Bulldozer compute modules
  - 16 cores total



Next gen core architecture

- Two cores per Bulldozer 'module'
  - One shared FP unit per module
  - Native FMA implementation
- New Instructions
  - AVX, FMA4, XOP, AES
- 2M L2 cache per module

Evolved Northbridge

- 16M L3 cache (total), 32M combined L2+L3
- 4 DDR3-1600 channels (total)
- 4 HyperTransport™ links (total)
  - 6.4 GT/S, 16 bit, 12GB/Sec bi-directional

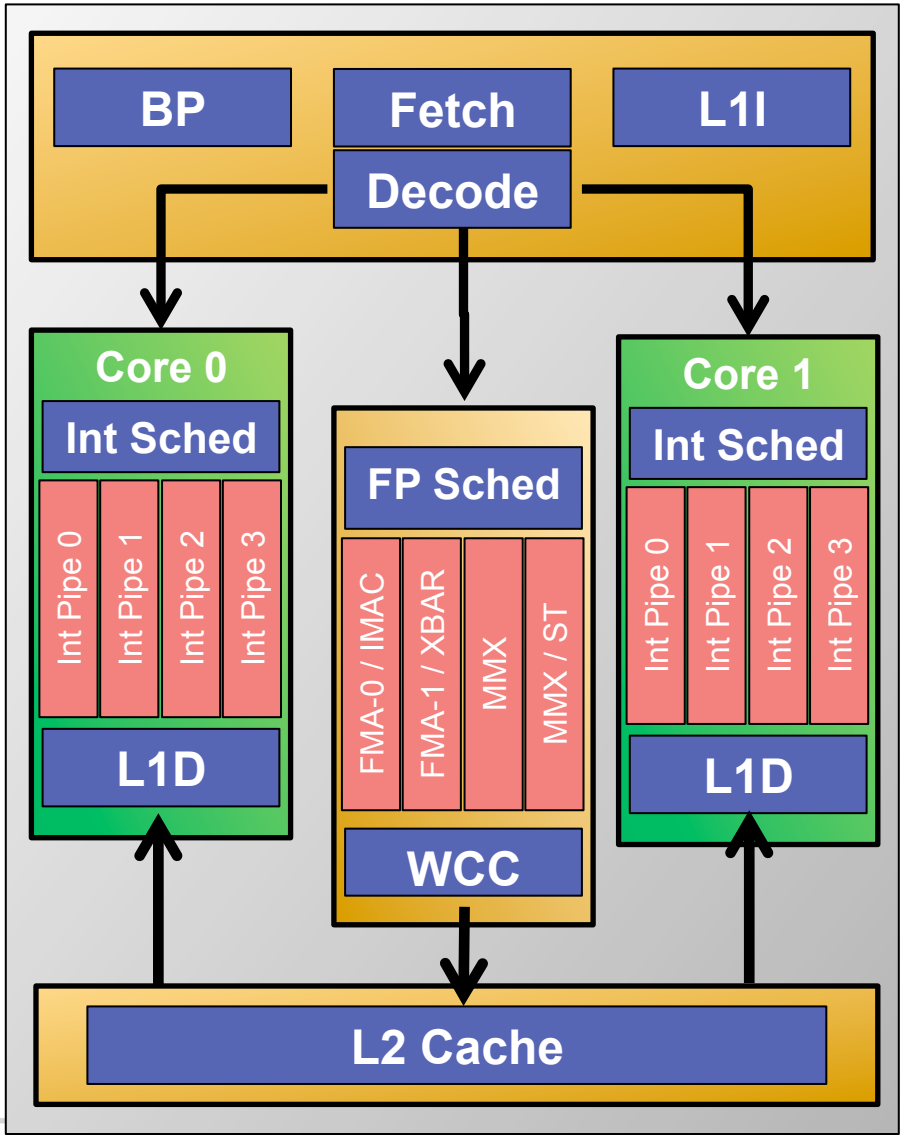
Simplified Bulldozer 2 Core Module

Multi-threaded Micro-architecture

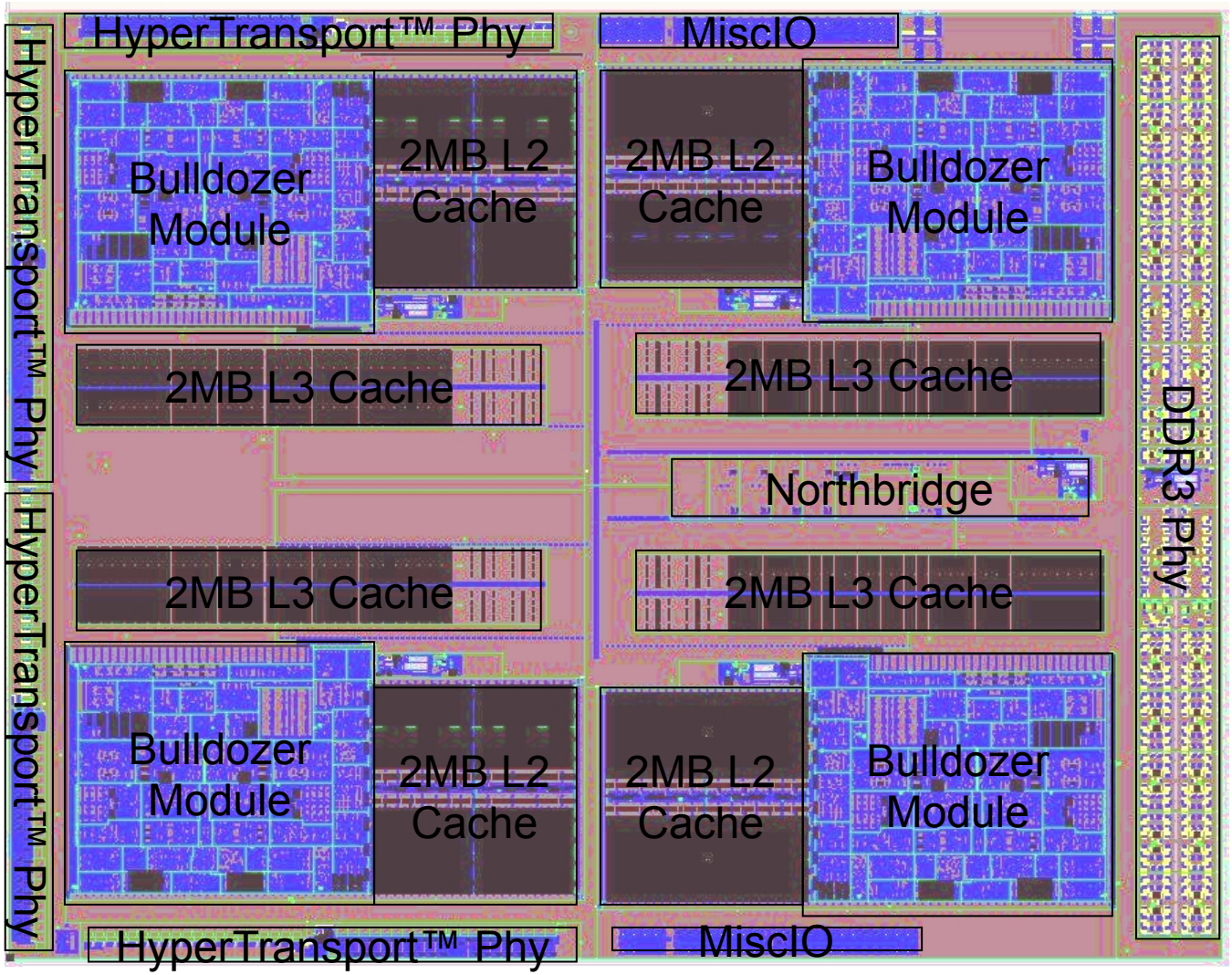
- Shared front end (IF, BP, DE)
  - Shared L1I
  - Shared Branch Prediction structures
  - Shared 4 way x86 decode
- Replicated Integer Cores
- Replicated L1D (with LD / ST Unit)
- Shared FPU
  - Amortize large AVX-256 unit
- Shared Write Coalescing Cache
- Shared L2 Cache

Flexible FP Unit

- Supports 1xAVX-256 or 2xSSE/AVX-128



Die Shot – Note High Re-use Design



Architecture Details

Cache Hierarchy

Shared L1I

64 KB, 2-way, 64 B line, 32 B fetch / clk

Dedicated Per Core L1D

16 KB, 4-way, 64 B line, Write Through
2 x 128 bit LD / clk , 1 x 128 bit ST / clk
4 clk LD to use latency
40 entry LD queue, 24 entry ST queue

Shared L2

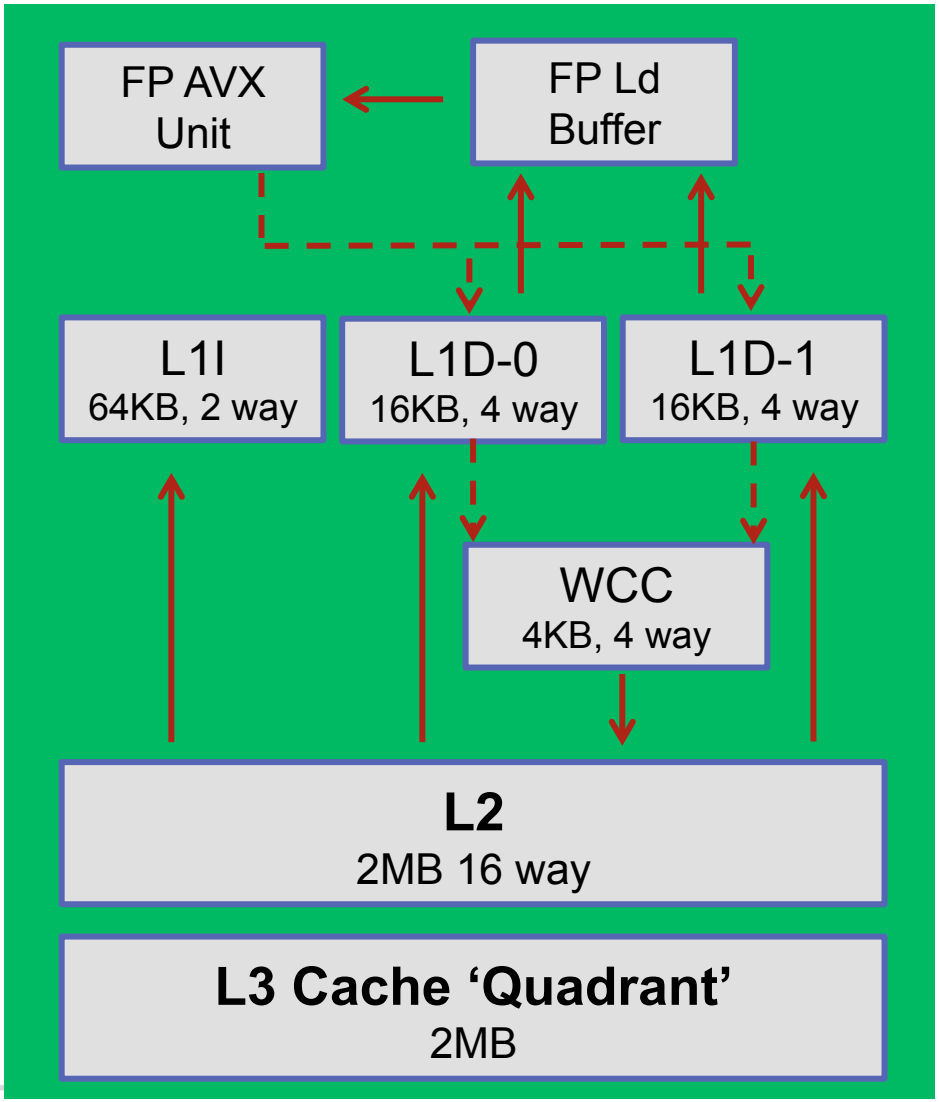
Unified 2MB, 16-way, 64 B line
20 clock LD to use latency
Up to 23 outstanding misses

WCC

4 KB, 4 way, coalesces stores

Distributed L3

4 x 2MB quadrants
Victim cache for L2, non-inclusive



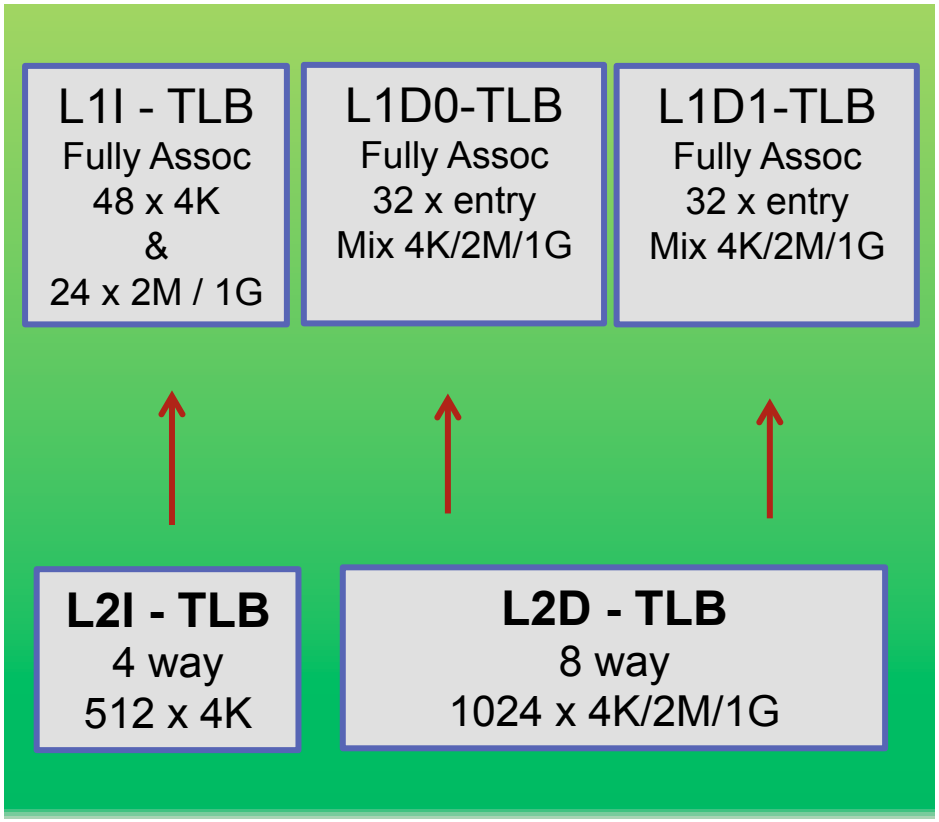
TLB Hierarchy

Shared L1 ITLB
- 48 x 4K pages & 24 x 2M or 1G
- Fully associative

Shared L2 ITLB
- 512 x 4K pages
- 4-way associative

Dedicated L1 DTLBs
- 32 x 4K or 2M or 1G
- Fully associative

Shared L2 DTLB
- 1024 x 4K or 2M or 1G
- 8-way associative



4M pages are also supported, but consume 2 x 2M entries

FPU Functional Unit and Pipe Layout

Execution

4 x 128 bit execution pipelines, 4 ops/clk

LD / ST

2 x 128 bit LD/Clk + 1 x 128 bit ST/Clk

FMA

2 x 128 bit FMA / Clk or 1 x 256 bit FMA / Clk
Native FMA hardware, 4 operand variety
2 x FDIV/FSQRT operate in FMA units

Pipe 0

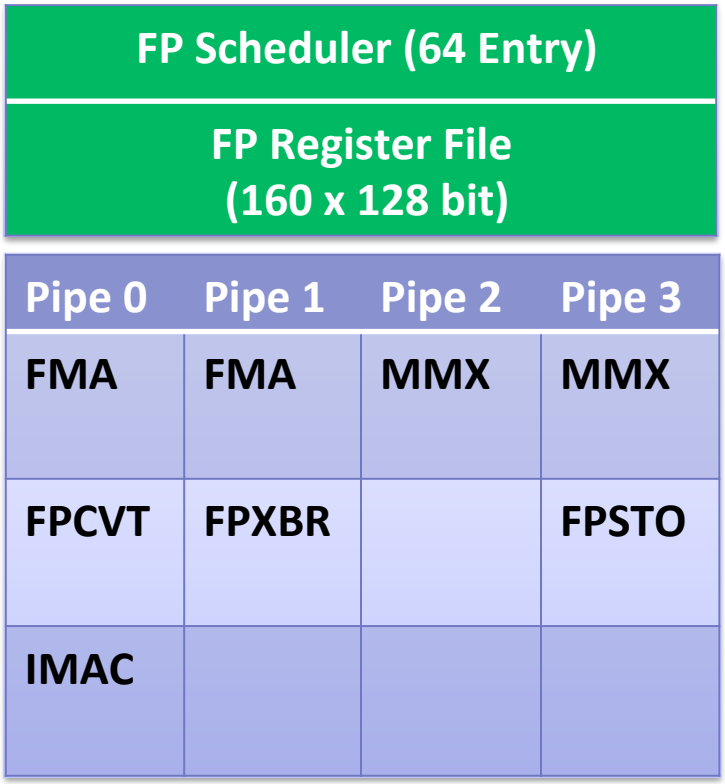
Implements 128 bit Integer MAC
Implements 128 bit FP conversion

Pipe 1

Crossbar unit for permute, shuffle, pack

Pipe 2, 3

Implement 128 bit integer ops



SSE vs AVX vs FMA4

SSE	AVX	FMA4
2 operands	3 operands	4 operands
a = a *op* b	c = a *op* b	d = a + b * c

SSE

Destroys operand a, so it is often saved to another register first

AVX

Requires two instructions to implement fadd(c, fmul(a, b))

FMA4

Single, non-destructive, instruction for fadd(c, fmul(a, b))

A Simple Test: x[i] = y[i] + z[i]

	1 Thread	2 Thread	4 Thread
L1	28	56	72
L2	11.5	23	28
Mem	3.5	5.5	5.7

Vector add using SSE via gcc vector types (v2df)

vector length is varied to fit into a particular cache size

This system has 2 modules, or 2 x FPUs

peak bytes / clk per FPU is 48 (2 x 16B LD + 1 x 16B ST)

table shows measured bytes / clk for the quad core system

Results for L1 / 4 Threads show

36 B / clk vs 48 B / clk peak

AVX-128 showed similar results, did not test AVX-256

Results for L2 need further investigation

These results are not validated or endorsed by AMD

Miscellaneous

HW Prefetcher

- Prefetch strided data into the L1 data cache
  - Recognizes a large number of streams & strides
  - Propagates through L2
- Prefetches non-strided but correlated data access patterns

Branch Predictor

- See Software Optimization Guide for more details
- BTB (Two levels, large number of entries)
- Hybrid predictor for conditionals
- Indirect predictor (object oriented code)
- RSA – 24 entry

HT Assist

- Dedicates part of L3 cache to a snoop directory

Miscellaneous

Instruction Latencies

See Software Optimization Guide for more details

Note that the listed latencies are best case / un-contended

New Instructions

AVX-128 & 256 – new instructions for packing & permute, faster decode

FMA4 – 4 operand fused multiply-add, ~10% lift on SPECfp_rate

AES / ABM / CLMUL – cryptography & related

XOP – horizontal integer ops

See Software Optimization Guide for more details

Time Stamp Counter (RDTSC)

Counts at the rated frequency of the CPU

BD SW Optimization Notes

A Short List

Cray has performance tuning experts . . .

- Use them and use the Cray tools

Cray has high performance math libraries

- Use them whenever possible

Memory Management

- NUMA Awareness is critical
  - Allocate local to a die, avoid remote memory access across HT links
- SIMD Alignment is still important
  - Unaligned SIMD instructions are available, however more complex uops

FMA4 & AVX-128

- Use FMA rather than fmul + fadd, max flop/clk & decrease issue pressure
- Use SSE* + FMA or AVX-128 + FMA rather than AVX-256

Tight Code

- Re-use registers as much as possible, unroll loops

Software Optimization Guide for AMD Family 15h

Roughly 350 pages, much aimed at compiler writers

The SWOG's Key List (Page 22)

Table 2. Optimizations by Rank

Rank	Optimization
1	Load-Execute Instructions for Floating-Point or Integer Operands (See section 5.1 on page 79.)
2	Write-Combining (See section 6.6 on page 111.)
3	Branches That Depend on Random Data (See section 7.3 on page 121.)
4	Loop Unrolling (See section 8.2 on page 129.)
5	Pointer Arithmetic in Loops (See section 8.5 on page 136.)
6	Explicit Load Instructions (See section 10.2 on page 168.)
7	Reuse of Dead Registers (See section 10.14 on page 184.)
8	ccNUMA Optimizations (See section 11.1 on page 193.)
9	Multithreading (See section 11.3 on page 204.)
10	Prefetch and Streaming Instructions (See section 6.5 on page 103.)
11	Memory and String Routines (See section 6.8 on page 113.)
12	Floating-Point Scalar Conversions (See sections 10.15 on page 185.)

developer.amd.com / docs / guides
pdf #47414, Rev 3.06 January 2012

Additional Hazards

Branch Mispredict – 20 clocks

Present in some heavily OO HPC codes

STLF – 20 clocks

Data from a store instruction is loaded by a subsequent instruction before retire

HW attempts to deal with this, but best to avoid this instruction sequence

STLI – smallish penalty

Related to STLF

L1D Bank Thrash

L1D has 16 banks, each 128 bits wide

Only one load from a given bank per clock

AVX-256 Pro and Con

There three 'modes' to operate a Bulldozer module

Run 2 threads, 2 x AVX-128 / SSE instruction streams
Run 2 threads, 2 x AVX-256 instruction streams
Run 1 thread, 1 x AVX-256 instruction stream

AMD recommends the first, 2 x AVX-128 / SSE

Can treat / load both cores symmetrically
Get 2x performance boost for scalar FP code

For the second mode, 2 x AVX-256

Won't be fetch limited in general , avoids some types of internal stalls
AVX-256 code has many fewer free rename registers (PRNs)
Machine may stall sooner due to lack of PRNs
AVX-256 may have dispatch restrictions
Some instructions are 'fastpath doubles' and must dispatch together
Only 1 x 128 bit store per clock

For the third mode, 1 x AVX-256

Less pressure on the rename registers vs 2nd mode
Doubles the effective cache size per thread – can be crucial

Vresid Tests

Vresid

Compute kernel, element force calculations for structural dynamics code

Very cache friendly

Originally from SNL code, kernel ran on ASCI Red at 25% of peak FP on Pentium Pro

Single precision calculations here, for test purposes only

Results Show (with all four cores loaded)

10% lift with fma

2.8x speedup for scalar code vs 128 bit SIMD

10% lift for fma + 128 bit SIMD vs fma + 256 bit SIMD

Code still hits 25% of peak FP

	Explicit Type SIMD Width	1 Task	2 Tasks	4 Tasks
Gcc4.4, no fma4, barcelona (Scalar SSE)	1	138 sec	76 sec	60 sec
Gcc4.6, fma4, bdver1 (Scalar SSE)	1	122 sec	70 sec	53 sec
Gcc4.6, fma4, bdver1 (Packed SSE)	4	43 sec	25 sec	19 sec
Gcc4.6, fma4, bdver1 (Packed AVX)	8	40 sec	23.5 sec	21 sec

These results are not validated or endorsed by AMD

Resources

Cray Performance Experts / Cray Tools

Cannot over-emphasize this

http://developer.amd.com

Software Optimization Guide for AMD Family 15h Processors

Rev 3.06, January 2012, pdf 47414, under docs / guides

Chapter 2 has a summary of the micro-architecture

Appendix B has instruction latencies, C has NUMA material

BIOS & Kernel Developer Guide

AMD64 Arch Programmer's Manual Vol 4: 128 & 256 bit Media Instructions

ACML (AMD Core Math Library)

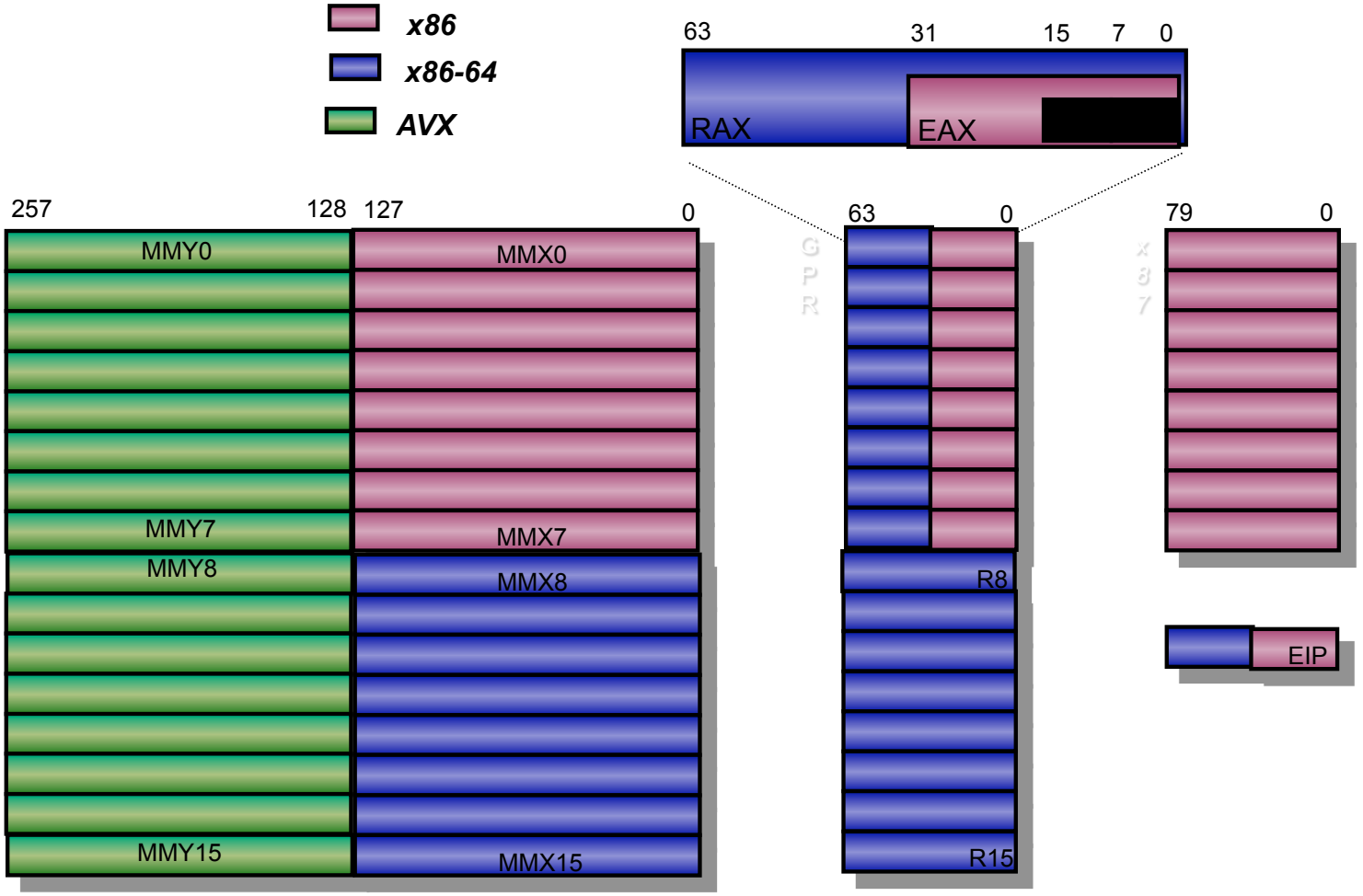
CodeAnalyst – profiling tool

References

(1,2) full information on HPL & Stream benchmark data can be found at http://www.amd.com/us/products/server/benchmarks/Pages/benchmarks-filter.aspx

Backup

User Visible Register Set



Compute Node: Cray Xe6 2 x G34 Sockets

2 x Interlagos CPUs

32 x Next Gen Bulldozer Cores

AVX, FMA4

2.* GHz base frequency

16M combined L2

12M combined L3

Non-inclusive hierarchy

8 x Memory Channels

DDR3-***, 1 x DPC

No I/O Hub on compute nodes