Software is Key

Intel MIC: GPU level performance with CPU level programmability

Jim Cownie <james.h.cownie@intel.com>
Intel

3rd UK GPU Computing Conference

Optimization Notice

Optimization Notice

Intel's compilers may or may not optimize to the same degree for non-Intel microprocessors for optimizations that are not unique to Intel microprocessors. These optimizations include SSE2, SSE3, and SSSE3 instruction sets and other optimizations. Intel does not guarantee the availability, functionality, or effectiveness of any optimization on microprocessors not manufactured by Intel. Microprocessor-dependent optimizations in this product are intended for use with Intel microprocessors. Certain optimizations not specific to Intel microarchitecture are reserved for Intel microprocessors. Please refer to the applicable product User and Reference Guides for more information regarding the specific instruction sets covered by this notice.

Notice revision #20110804

Legal Disclaimer

http://intel.com/software/products

Who am I?

Overview

- History
  - Software
  - Benchmarking
  - Hardware
- Hardware Fundamentals
  - Moore's Law
  - Energy
- Where are we now
  - Xeon
  - MIC
- Conclusions

History: Software
How old is HPC software?

Code	Area	Date
NASTRAN	Structures	1968
Spice	E-Cad	1972
Pam-Crash	Structures	1978
UKMO Unified Model	Weather	1990
PETSc	Solvers	1991
LAPACK	Solvers	1992
NWCHEM	Chemistry	1995
WRF	Weather	2000

- Code lasts much longer than hardware
- We must support old code on new hardware

History: Software
How old are languages?

Language	Date
Fortran	1966 (FORTRAN 66)
C	1978 (K&R)
C++	1985 (C++ Programming Language)
MPI	1994
OpenMP	1997

- Languages that work have a long life
  - Investment in code
  - Investment in brain-cells
- All have open specifications & many implementations
  - Formal standards (C, C++, Fortran)
  - Open industry standards (MPI, OpenMP)
- We must support old languages on new hardware

History: Old benchmarking tricks come around too…

How do you make an accelerator look good?

- Only time kernels, not the whole code
  - Even if I can offload 25% of the execution and have it run infinitely fast, that's still only a 1.33x speedup
- When quoting speedup ignore data transfer time
  - or choose a benchmark with no data (e.g. Mandelbrot)
- Change the algorithm but don't use the new one on the CPU
  - (Special case) Compare single precision results on the accelerator with double precision on the CPU
- Compare a newly released accelerator with a two year old CPU

History: Old benchmarking tricks come around too…

What's the easiest way to get a good speedup?

- Start with something slow…
- So don't optimize the CPU case
  - Use old or poor compilers and compile for an 8087
  - Only use a single core on the CPU even if it has twelve or more
  - Spend all your effort on the accelerator code
  - Assume that effort to rewrite code for the accelerator is free, but that there is no effort to do any tuning of the CPU code

History: Hardware

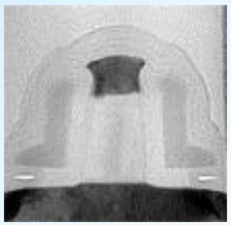
Hardware	Flops	Date
FPS AP-120B	12 M	1976
Intel 8087	50 K	1980
ClearSpeed	25 G	2003

- The wheel of reincarnation turns
- Computing at the end of an I/O bus is **hard**
- Implementations which work end up on the CPU die
  - Moore's law; we need something to use all the transistors profitably
  - 8087 moved into the CPU
  - SIMD vector floating point moved into the CPU (SSE, SSE2, AVX, AVX2, …)
  - Crypto instructions moved to the CPU (SPARC and Xeon AES-NI)

Fundamentals:
Moore's law is alive and well

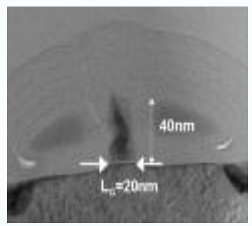
New Intel technology generation every 2 years.
Intel R&D technologies drive this pace well into the decade.

90nm
2003

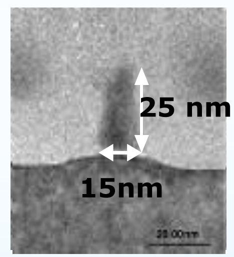


65nm
2005

45nm
2007

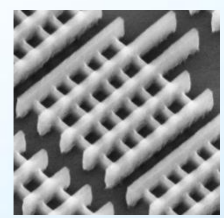


32nm
2009



Hi-K metal-gate

22nm
2011



3-D Tri-gate

14nm
2013

10nm
2015

Shrink

We will have lots of transistors!

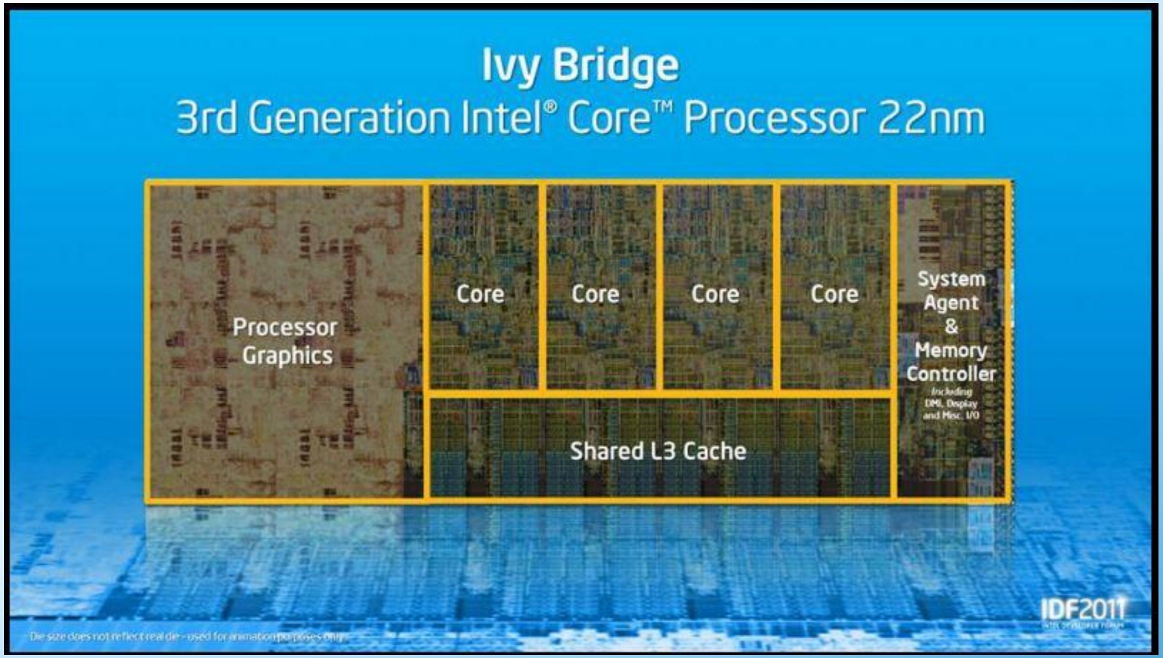
Fundamentals: Energy

Nvidia* energy numbers	2010 40nm	2017 10nm high freq	2017/2010
DP FMA	50 pJ	8.7 pJ	17%
3x64bit read, 1x64bit write in 8K SRAM	56 pJ	9.6 pJ	17%
Wire energy (256b,10mm)	310 pJ	200 pJ	65%
DRAM Access	10,000 pJ	1,700 pJ	17%

- ALU ops themselves are cheap
- Locality (even on die) is important, and becomes critical in the future
  - Caches matter
  - A streaming architecture will not scale forward

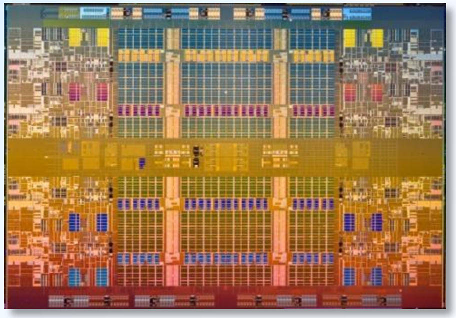
Where are we now?

- Hardware Convergence
  - Mainstream chips have multiple cores and SIMD vector units (AVX: 256 bits (8 float,4 double); MIC: 512 bits)
  - GPU hardware is acquiring normal CPU features so that it is easier to program
    - Caches
    - Subroutine calls
- Accelerators move on die
  - Off die is too slow
  - Accelerator should be in process’s address space
  - We have all those transistors to use

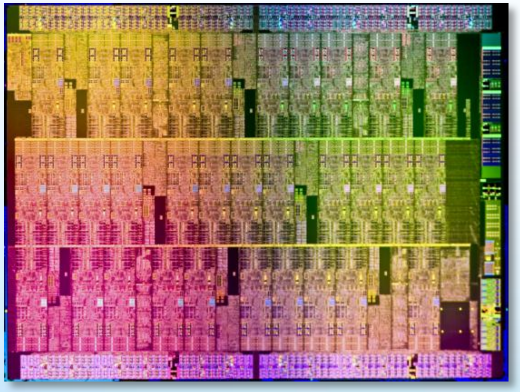


Intel's Many Core and Multi-core Engines

Die Size not to scale



Multi-core Intel® Xeon® processor at 2.26-3.5 GHz



Many Integrated Cores at 1-1.2 GHz

Intel Xeon processor:

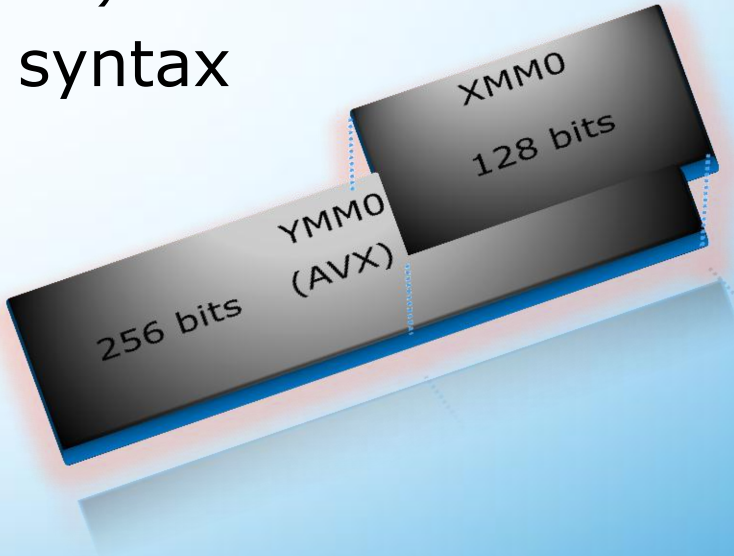
- Foundation of HPC Performance
- Suited for full scope of workloads
- Industry leading performance/watt for serial & parallel workloads.

MIC processor:

- The performance of a highly parallel processor
- The benefits of familiar, standard programming models

Xeon: Intel® Advanced Vector Extensions (Intel® AVX)

- 2X the throughput of SSE
- Extend SSE FP instruction set to 256 bits operand size
  - Intel AVX extends all 16 XMM registers to 256bits (8 single-precision, 4 double-precision)
- New, non-destructive source syntax
  - VADDPS ymm1, ymm2, ymm3
- New Operations to enhance vectorization
  - Broadcasts
  - Masked load & store



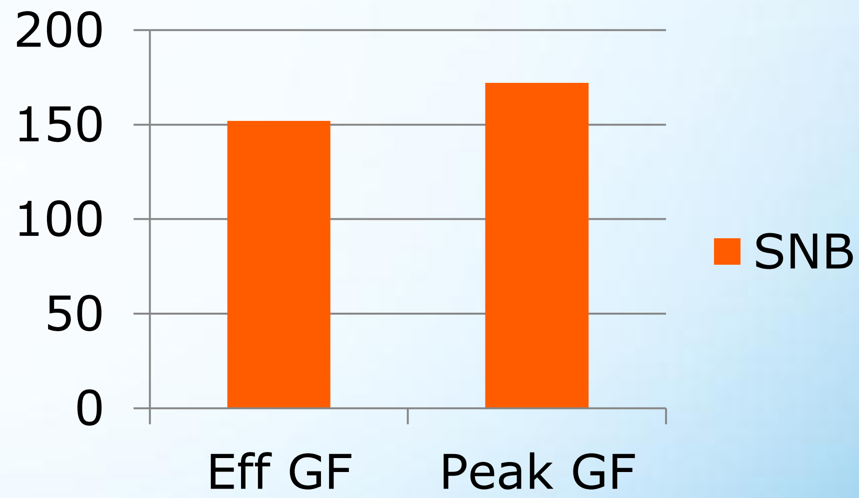
Xeon: Breakthrough Performance

Intel® Xeon® Processor E5 Family

Max 152GF effective flops per socket, 91% efficiency on HP Linpack

172 peak Gflops / socket – 2x improvement with Intel® AVX

Industry's first integrated PCI Express* 3.0



Nov 2011 Top500 listing
#105 SNB cluster

Source: Top500.org, November 2011

MIC: Knights Corner

- Exists in 22nm process
- > 50 cores/die
- 512 bit SIMD instructions
- Early Si delivers 1TFlop sustained on DGEMM
- Runs Linux
- Can be
  - a network node (ssh in…)
  - used as an offload processor over PCIe
- Targeted by Intel compilers

Invest in Common Tools and Programming Models

Multicore

Intel® Xeon® processors are designed for intelligent performance and smart energy efficiency

Continuing to advance Intel® Xeon® processor family and instruction set (e.g., Intel® AVX, etc.)

Code

Use One Software Architecture

Today

Tomorrow

Many-core

Intel® MIC Architecture - co-processors are ideal for highly parallel computing applications

Software development platforms ramping now

Use One Software Architecture Today. Scale Forward Tomorrow.

ORNL & Univ. of Tennessee on Intel® MIC Architecture:

Experience with Knights Ferry design and development kit

MIC Partner

- Unparalleled productivity: in under 3 months
  - Ported all of NWChem (chemistry), ENZO (astro.), ELK (mat. sci.), MADNESS (app. math.), MPI, GA, …
  - Correct ports in less than one day each
  - Circa 5M LOC (Fortran 77/90, C, C++, Python)
  - MPI, Global Arrays, …
- Most of this software does not run on GPGPUs and probably never will due to cost and complexity
- Demonstrated execution modes:
  - Native mode: KNF is fully networked Linux system
  - Offload mode: KNF is an attached accelerator
  - Reverse offload mode: KNF in native mode offloads to host
  - Cluster mode: parallel application distributed across multiple KNF and hosts using MPI

NICS

MIC: Achieving Performance

- Parallelize code
  - Reduce serial code as much as possible
  - Minimize critical sections
  - Improve load balance (or use a model that handles it)
- Vectorize code
  - Loop transformations
  - Vectorization directives/pragmas
  - Reductions
- Memory hierarchy optimizations
  - Blocking, Cache-Oblivious algorithms, …
- Use existing tools on Xeon
- All of these benefit code on CPUs as well
  - Doing them now is worthwhile
  - MIC is just more “extreme”, or “focused”

Depressing Conclusions

- There are no silver bullets
- Data parallelism & vectorization is important for all of the current hardware
  - and will become more important in the future
- As it always has, new hardware requires tuning to achieve performance
  - Unfortunately, using the same programming language doesn't mean performance is portable
- As a community we forget our history and love the new
  - Vectorization is "so 1970's" no-one would publish a paper on that nowadays
  - Papers on tuning are hard to publish, yet it's easy to publish papers on rewriting code in "language du jour"

Cheerful Conclusions

- There are no silver bullets
  - We'll all be in work for a while yet!
- You don't have to use new programming languages
  - Fortran, C, C++,… can all be used
  - Parallelism can be expressed with OpenMP, TBB, Cilk™Plus, pthreads (if you must), MPI, …
- Improving code for data and thread parallelism is not wasted work
  - It pays off on the CPU as well as on accelerators

Backup

- There is much more public information on MIC available, for instance
  - Highly Parallel Applications and Their Architectural Needs
  - Fast Sort on CPUs, GPUs and Intel MIC Architectures
- Remember, Google Is Your Friend.

Other benchmark cheats

- Less relevant for GPU, but for parallel…
  - Compare internal speedups, not absolute speeds.

Amdahl was Smart

- Amdahl's law is useful for thinking about accelerators as well as parallelism
  - Treat the accelerated portion of the code as if it were parallel
  - Use the accelerator kernel speedup as the parallel section speedup ("Nprocessors")
  - Out pops the overall speedup
  - For a better estimate include the data transfers
  - Easily gives "speed of light" numbers (set offload kernel time to zero)

References/Notes(for PDF version)

Slide 4: Picture from http://en.wikipedia.org/wiki/File:Daniellion.jpg “This image (or other media file) is in the public domain because its copyright has expired.
This applies to Australia, the European Union and those countries with a copyright term of life of the author plus 70 years.”

Slide 6: Dates are as close to release 1.0 as I can find. Sources below

- http://en.wikipedia.org/wiki/LAPACK
- http://onlinelibrary.wiley.com/doi/10.1002/qua.560560851/abstract
- http://en.wikipedia.org/wiki/Pam-Crash
- http://www.metoffice.gov.uk/research/modelling-systems/unified-model
- http://www.mmm.ucar.edu/mm5/mpp/ecmwf01.htm
- http://www.mcs.anl.gov/petsc/documentation/tutorials/Speedup10.pdf

Slide 12: Energy numbers from Nvidia

- http://research.nvidia.com/publication/gpus-and-future-parallel-computing page 9

Slide 13: Die picture shows that the CPU is not the largest thing on the die. Compare the area of the graphics with the four cores. Total graphics ~= Total Cores.

Cores are certainly < 50% of the die.

Slide 15: Details at http://software.intel.com/en-us/avx

Slide 19: From NICS SC11 presentation, video at http://www.youtube.com/watch?v=TOVokMClr5g makes the same points but doesn’t include this exact slide…

Slide 20: Programming models that handle imbalance are things like Cilk™Plus or TBB, as against OpenMP’s default static scheduling