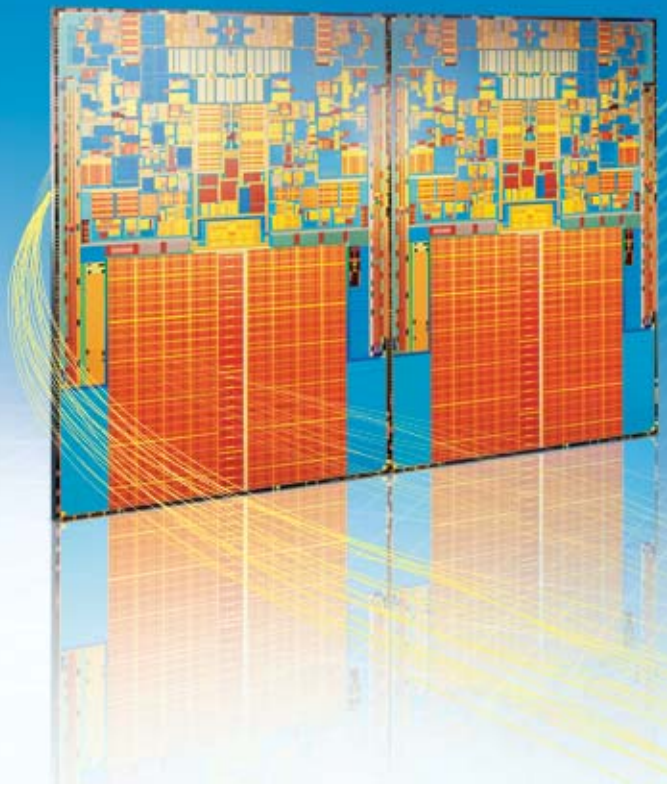

Opportunities with OpenCL

Aaron Lefohn
Larry Seiler

Intel Corporation

Agenda

- **OpenCL for graphics**
- Heterogeneous parallel coordination in OpenCL
- Data parallelism in OpenCL
- Task parallelism in OpenCL
- Conclusion

Evolution of Graphics Programming

- OpenCL part of the evolution of interactive graphics to "fully programmable graphics" (i.e., software rendering)

- Papers at SIGGRAPH 2009 that mix compute/pipeline
  - Gaussian KD-Trees for Fast High-Dimensional Filtering
  - An Efficient GPU-based Approach for Interactive Global Illumination
  - Hardware-Accelerated Global Illumination by Image Space Photon Mapping (HPG)
  - …

- Beyond Programmable Shading course
  Thursday, 8:30am – 5:30pm, Auditorium A

Agenda

- OpenCL for graphics
- **Heterogeneous parallel coordination in OpenCL**
- Data parallelism in OpenCL
- Task parallelism in OpenCL
- Conclusion

OpenCL is a Parallel Coordination API

- OpenCL's "host" API coordinates heterogeneous parallel computations
  - Literature rich with parallel coordination languages/API
  - OpenCL unique in its ability to coordinate CPUs, GPUs, etc
- Key coordination concepts
  - Each device has its own asynchronous workqueue
  - Synchronize between OCL computations w/event handles from different (or same) devices
  - Enables algorithms and systems that use all available computational resources
  - Enqueue "native functions" for integration with C/C++ code

Agenda

- OpenCL for graphics
- Heterogeneous parallel coordination in OpenCL
- **Data parallelism in OpenCL**
- Task parallelism in OpenCL
- Conclusion

OCL's Two Styles of Data-Parallelism

- Implicit (shader-style) data parallelism
  - Write the kernel as if it were a scalar program
  - Use vector data types sized naturally to the algorithm
  - Kernel automatically mapped to SIMD compute resources by compiler and/or hardware

- Explicit-vector-width data parallelism
  - Write the kernel using wide vector types
  - Use vector types sized to match native HW width
  - Use in task parallelism or optimizing to a particular HW

OCL's Two Styles of Data-Parallelism

- Data parallel kernels on Intel architectures
  - Either method work on CPUs, GPUs, Larrabee, …
    - SSE/AVX/LRBni: 4/8/16 workitems in parallel
  - Hybrid use of the two methods
    - AVX: can run two 4-wide workitems in parallel
    - LRBni: can run four 4-wide workitems in parallel

- How to use explicit method in OpenCL
  - Programmer chooses vector data type (width)
  - Compiler hints using attributes
    - vec_type_hint(type*n)*

Agenda

- OpenCL for graphics
- Heterogeneous parallel coordination in OpenCL
- Data parallelism in OpenCL
- **Task parallelism in OpenCL**
- Conclusion

Task Parallelism in OpenCL

- clEnqueueTask
  - Imagine “sea of different tasks” executing concurrently
  - A task “owns the core” (i.e., a workgroup size of 1)

- Use tasks when algorithm…
  - Benefits from large amount of local/private memory
  - Has predictable global memory accesses
  - Can be programmed using explicit vector style
  - “Just doesn’t have 1000’s of identical things to do”

- Use data-parallel kernels when algorithm…
  - Does not benefit from large amounts of local/private memory
  - Has unpredictable global memory accesses
  - Needs to apply same operation across large number of data elements

Task Parallelism in OpenCL

- Host
  - `clEnqueueTask`
- Device
  - Use `kernel_exec(1, typeN)` instead of `kernel`

Agenda

- OpenCL for graphics
- Heterogeneous parallel coordination in OpenCL
- Data parallelism in OpenCL
- Task parallelism in OpenCL
- **Conclusion**

Conclusions

- OpenCL is part of evolution of interactive graphics toward software rendering
- OpenCL coordinates a heterogeneous set of CPUs, GPUs, and other processor types
- OpenCL supports two forms of data parallelism that both map well to Intel architectures
- OpenCL supports task and data parallelism: “Use the right tool for the job”