Real-Time Shallow Water Simulation with OpenCL for CPUs

Adam Lake, Arnon Peleg
software, Intel
OpenCL WG, The Khronos Group
Acknowledgements

- Demo: Dmitry Budnikov, Konstantin Rodyushkin, Alexei Klishin, Alexei Rukhlinskiy, Maxim Shevstov
- Marketing: Arnon Peleg
- Prior Content: Ofer Rosenberg/Tim Mattson
- Art Assets: Glen Lewis, Jeffery Williams
CPUs, OpenCL, Heterogeneous Computing

- **OpenCL is a Platform API** which supports a uniform programming environment across devices:
  - Enables heterogeneous parallel computations
  - Unique in its ability to coordinate CPUs, GPUs, etc

- **Make the best use of all available resources (CPU’s, GPU’s) from within a single program:**
  - One program that runs well (i.e. reasonably close to “hand-tuned” performance) on a heterogeneous mixture of processors.
  - 2nd Generation Intel® Core™ Processor Family: a new level of integration between CPU & GPU
Writing OpenCL for the CPU

- OpenCL can be used to harness potential of any CPU
  - Humanly readable vectorized source (like shaders!)
  - Our results indicate close to hand tuned performance with our current generation OpenCL C compiler
    - *Getting better all the time!*
  - Forward compatibility from one CPU Generation to the next
  - Cross vendor portability
  - Code maintainability
  - Code readability
How does OpenCL map to the CPU?

* Taken from OpenCL 1.1 Specification, Rev 33
Mapping OpenCL Data Parallel Execution Model to SIMD

- **Implicit (common case)**
  - Easy enough, just like writing shaders!
  - Write kernel as scalar and vectors that map naturally to workloads
  - Compiler handles mapping from scalar to vector
  - Hint: Experiment with `–cl-fast-relaxed-math` flag for increased perf
  - *Good for game developers: accuracy vs. perf tradeoff*

- **Explicit SIMD data parallelism**
  - Kernel defines single stream of instructions for SIMD Unit
  - Vector size matches hardware width
  - Programmer can use a hint on the kernel
    - `vec_type_hint(typen)`
    - *If it matches machine SIMD width then explicit*

- **See OpenCL 1.1 Spec for more details**
Overview of Vectorization

Vectorization enables developer to exploit the CPU Vector Units in Implicit Data Parallelism

Vectorizing code…

Reduced number of Vector invocations

© Copyright Khronos Group, 2010
OpenCL in the Shallow Water Demo
Shallow Water Example

Uses Flux splitting method for solving Navier Stokes equations:

\[
\begin{align*}
\frac{\partial H}{\partial t} + \text{div}(Hw) &= 0 \\
\frac{\partial Hw}{\partial t} + \text{div}(Hww) + \frac{g}{2} \text{grad}H^2 &= Hg \text{grad}d \\
\end{align*}
\]

- \( H \) is fluid depth
- \( w \) is fluid velocity vector
- \( G \) is gravitational acceleration constant
- \( d \) is water depth measured from still water surface

- This talk focuses on lessons learned mapping to OpenCL
- See References for more details on the algorithm
- Sample expected to be part of Intel OpenCL SDK
- Entire simulation \(~1000\) lines OpenCL C
```cpp
void CFlux2DScalarCalculator::CalcV_Pos_MoreCritical()
{
    // Load values for the up node
    float H_ = inPtrUp.pH[0];
    float u_ = inPtrUp.pU[0];
    float v_ = inPtrUp.pV[0];
    float Hb_ = pHbUp[0];

    // First do the calculations with the center (H, u,v) point in the grid
    float tmpFlowH = cur.H * cur.v;
    float tmpFlowU = cur.H * cur.v * cur.u;
    float tmpFlowV = cur.H * (halfGravity * cur.H + cur.v * cur.v);

    // Calculate the positive portion of the water flow
    tmpFlowH -= H_ * v_;
    tmpFlowU -= H_ * v_ * u_;
    tmpFlowV -= H_ * (halfGravity * H_ + v_ * v_);

    // In the final formula, Ru is taken with a minus
    tmpFlowV -= halfGravity * (cur.H + H_) * (curHb - Hb_);

    // For the final step, write the results to the flow outputs
    flow.H += tmpFlowH * tau_div_deltah;
    flow.u += tmpFlowU * tau_div_deltah;
    flow.v += tmpFlowV * tau_div_deltah;
}
```
From C to CL

```c
void CalcV_Pos_MoreCritical(CFlux2DLocalPtrs *pBuffers, __global OCLKernelData *pData) {
    // Load values for the up node
    float H_ = pBuffers->inPtrUp.pH[0];
    float u_ = pBuffers->inPtrUp.pU[0];
    float v_ = pBuffers->inPtrUp.pV[0];
    float Hb_ = pBuffers->pHbUp[0];

    // First do the calculations with the center (H, u, v) point in the grid
    float tmpFlowH = pBuffers->cur.H * pBuffers->cur.v;

    // Calculate the positive portion of the water flow
    tmpFlowH -= H_ * v_;
    tmpFlowU -= H_ * v_ * u_;
    tmpFlowV -= H_ * (pData->halfGravity * H_ + v_ * v_);

    // In the final formula, Ru is taken with a minus
    tmpFlowV -= pData->halfGravity * (pBuffers->cur.H + H_) * (pBuffers->curHb - H_);

    // For the final step, write the results to the flow outputs
    pBuffers->flow.H += tmpFlowH * pData->tau_div_deltah;
    pBuffers->flow.u += tmpFlowU * pData->tau_div_deltah;
    pBuffers->flow.v += tmpFlowV * pData->tau_div_deltah;
}
```

"The most complex task is passing parameters which were encapsulated in [a] separate class in [the] original C++ version of [the] solver"

Dmitry Budnikov, iNNL
Relative solver Performance within same grid size

Results measured on Core™ i7 975, 3.3 GHz, 6GB DDR3

Results depend on the algorithm/implementation

Use relaxed math flag when possible with OpenCL!

Game Dev Sweet Spot!

- Native C Serial (Single-threaded)
- Native SSE Serial (Single-threaded)
- Native C OpenMP (Multi-threaded)
- OpenCL
- OpenCL (Relaxed math)
- Native SSE OpenMP (Multi-threaded)
- REFERENCE PERF. (Native C serial)

1 Results measured on Core™ i7 975, 3.3 GHz, 6GB DDR3
2 Results depend on the algorithm/implementation
‘FPS’ performance w/ no rendering

Sweet Spot!

1 Results measured on Core™ i7 975, 3.3 GHz, 6GB DDR3
2 Results depends on the algorithm/implementation
Call to Action

• See the demo in action!
• Download the SDK(s)
  - software.intel.com/en-us/articles/intel-opencl-sdk/

• Give feedback to hardware vendors
• *Give feedback to OpenCL Working Group on improvements you want to see in OpenCL, the industry standard for heterogeneous computing!*
References

- s09.idav.ucdavis.edu for slides from a Siggraph2009 course titled “Beyond Programmable Shading”
- Lake, A., Game Programming Gems 8, General Purpose Computing on GPUs, Chapter 7.
- Stocker J., Waves in Water [Russian translation], IL, Moscow (1959).
- Grigoriev B., Belyaev V., Differential scheme of splitting vector flows for shallow water equations, Saint Petersburg.
Legal Disclaimer

- INFORMATION IN THIS DOCUMENT IS PROVIDED IN CONNECTION WITH INTEL® PRODUCTS. NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. EXCEPT AS PROVIDED IN INTEL’S TERMS AND CONDITIONS OF SALE FOR SUCH PRODUCTS, INTEL ASSUMES NO LIABILITY WHATSOEVER, AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO SALE AND/OR USE OF INTEL PRODUCTS INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT. INTEL PRODUCTS ARE NOT INTENDED FOR USE IN MEDICAL, LIFE SAVING, OR LIFE SUSTAINING APPLICATIONS.

- Intel may make changes to specifications and product descriptions at any time, without notice.
- All products, dates, and figures specified are preliminary based on current expectations, and are subject to change without notice.
- Intel, processors, chipsets, and desktop boards may contain design defects or errors known as errata, which may cause the product to deviate from published specifications. Current characterized errata are available on request.
- Any code names featured are used internally within Intel to identify products that are in development and not yet publicly announced for release. Customers, licensees and other third parties are not authorized by Intel to use code names in advertising, promotion or marketing of any product or services and any such use of Intel’s internal code names is at the sole risk of the user.
- Performance tests and ratings are measured using specific computer systems and/or components and reflect the approximate performance of Intel products as measured by those tests. Any difference in system hardware or software design or configuration may affect actual performance.
- Intel, Intel Inside, the Intel logo, and Intel Core are trademarks of Intel Corporation in the United States and other countries.
- OpenCL is trademarks of Apple Inc. used by permission by Khronos.
- *Other names and brands may be claimed as the property of others.
- Copyright © 2011 Intel Corporation. All rights reserved.
Optimization Notice

Intel® compilers, associated libraries and associated development tools may include or utilize options that optimize for instruction sets that are available in both Intel® and non-Intel microprocessors (for example SIMD instruction sets), but do not optimize equally for non-Intel microprocessors. In addition, certain compiler options for Intel compilers, including some that are not specific to Intel micro-architecture, are reserved for Intel microprocessors. For a detailed description of Intel compiler options, including the instruction sets and specific microprocessors they implicate, please refer to the “Intel® Compiler User and Reference Guides” under “Compiler Options.” Many library routines that are part of Intel® compiler products are more highly optimized for Intel microprocessors than for other microprocessors. While the compilers and libraries in Intel® compiler products offer optimizations for both Intel and Intel-compatible microprocessors, depending on the options you select, your code and other factors, you likely will get extra performance on Intel microprocessors.

Intel® compilers, associated libraries and associated development tools 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 Intel® Streaming SIMD Extensions 2 (Intel® SSE2), Intel® Streaming SIMD Extensions 3 (Intel® SSE3), and Supplemental Streaming SIMD Extensions 3 (Intel® 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.

While Intel believes our compilers and libraries are excellent choices to assist in obtaining the best performance on Intel® and non-Intel microprocessors, Intel recommends that you evaluate other compilers and libraries to determine which best meet your requirements. We hope to win your business by striving to offer the best performance of any compiler or library; please let us know if you find we do not.

Notice revision #20101101
backup
Implicit Data Parallelism on CPU

- One workitem per CPU lane
- Workitems packed into SSE registers
- Vector data types inside the kernel are made scalar and mapped to a single SSE lane
  - Simplifies optimization pass
- Vectorizer generates a workgroup (WG)
  - [optimizations]
- Kernel is executed over an ND Range divided into WG
- WG executed on compute unit
- Several workgroups run concurrently on all of the compute units
Explicit Data Parallelism on CPU

- Individual workitems executed on a single compute unit
- Vector ops mapped to SSE Units
- Vectors wider than physical registers are partitioned

- Workgroups (WG) executed on a single compute unit
- Barrier and fences impose penalties (context saving)

- Kernel is executed over an NDRange which is divided into WGs
- Several workgroups run concurrently on all compute units
One Member Company Slide

• You are allowed to do ONE slide on your company & products
• It can go first at the beginning after the title slide
• Say good stuff about your company
  - Use your own corporate template on this ONE SLIDE ONLY
• More good stuff about your company

Your logo

Buy Our Stuff
“The use of graphics, video, imaging, audio and parallel processing to enable visually and computationally intensive applications”
Using Slide Layouts

- If you have a large diagram of picture – considering putting it on the right
  - And using the shortened text box layout
Example Slide

- Make sure to put the title and main text into the correct template slots
- The slide format is intended to be used without extra blank lines
  - And for level 2 test to be used for sub-bullets
- See – you don’t need a blank line 😊
Example Slide

- Use images at every few slides
  - Keeps it more interesting
  - Keep your bullet points short