CLU: Open Source API for OpenCL Prototyping

Presenter: Adam Lake@Intel
Lead Developer: Allen Hux@Intel
Contributors: Benedict Gaster@AMD, Lee Howes@AMD, Tim Mattson@Intel, Andrew Brownsword@Intel, others
8/5/2012
Why CLU?

- OpenCL meant to be a low level, performance oriented API
- *But we hear you!*  
  - Getting started with OpenCL can be challenging...  
    - Devices, Queues, Contexts, Buffers, Platforms, Memory Objects, oh my!  
    - Cross industry effort to make it easier
Agenda

- CLU: What, When and Why
- CLU API Basic Functions
- CLU Generator
Simplifies 3 things

1. Program Initialization
2. CL kernel code compilation
3. Calling kernels with their arguments
#include <cl\cl.h>

int main()
{
    cl_platform_id platform = platforms[0];
    // normally load from file
    const char* prog =
        "kernel void Simple"
        "(int param1, global int* param2)"
        "{param2[get_global_id(0)] = param1;};";
    cl_ monstrates:
    cl_program p = clCreateProgramWithSource(m_context, 
        1, &prog, 0, &status);
    if (CL_SUCCESS != status) return status;
    status = clBuildProgram(p, m_numDevices, m_device, 
        m_buildOptions, 0, 0);
    if (CL_SUCCESS != status) return status;
    // create my buffer
    int bufferSize = sizeof(int)*bufferLength;
    cl_mem buf = clCreateBuffer(CLU_CONTEXT, 
        CL_MEM_WRITE_ONLY, bufferSize, 0, 0);
    // get a function that will enqueue my kernel
    clu_Simple s = cluCreate_Simple(0);
    // enqueue my kernel
    clu_enqueue_params params = CLU_DEFAULT_PARAMS;
    params.nd_range = (CLU_ND1(bufferLength));
    status = cluEnqueue_Simple(s, &params, 42, buf);
    // check the results
    int* mapped = (int*)clEnqueueMapBuffer(CLU_DEFAULT_Q, 
        buf, CL_TRUE, CL_MAP_READ, 0, bufferSize, 0, 0, 0);
    cluEnqueueUnmapMemObject(CLU_DEFAULT_Q, 
        buf, mapped, 0, 0, 0);
    cluReleaseMemObject(buf);
    // done using CLU
    cluRelease();
}
When to use CLU

- Think GLUT or DXUT

- Getting started

- Teaching

- Prototyping before integration
  - In *some* scenarios, deployment
    - within a studio, for example

- When you don’t want to think about low level details
  - Yes: University or lab environment
  - No: Probably wouldn’t use for cross OS, cross platform, highly configurable and chaotic consumer deployment

*Always allows you to drop down to low level OpenCL API*
CLU is not

- Not same source (not like C++ AMP or CUDA)
- Not a ‘wrapper’ for OpenCL objects
  - Directly uses OCL objects that you can use too
- Does not extend the kernel language
  - Cross platform with standard compliant OCL implementations
- Does not provide an intermediate representation
Why open source?

• Transparency—see how it works, no magic
• OpenCL is deployed on many platforms by many vendors
  - Too many for one member to maintain and track
• We needed a starting point, recognize our needs may be different than your needs
  - Hack away!
• Contribute, it’s YOUR codebase too!
Package Contents

• CLU is 3 files!
  - clu_runtime.h
  - clu_runtime.cpp
  - and clu_generator.cpp

• We include all build infrastructure for MS Visual Studio
• Documentation
• Other Build Environments coming
CLU in Visual Studio

Documentation!
Agenda

• CLU: What, When and Why
• CLU Basic Functions
• CLU Generator
Basic CLU functions

1. **Basic APIs:**
   - cluInitialize()
   - cluRelease()
   - cluBuildSourceFromFile()
   - cluEnqueue()
   - ...

2. **Request CL objects for Queues, Contexts, Devices**
3. **NDRange helpful macros**
4. **Utilities for Platform and Device Info**
Agenda

• CLU: What, When and Why
• CLU Basic Functions
• CLU Generator
Runtime Code Generator

1. Takes as input your kernel code (.cl file)
2. Generates a separate header file for your OpenCL application
   - Kernel source.
   - A function to create a cl_kernel object
     clugCreate_MyKernelName(…)
   - A function to enqueue the kernel
     clugEnqueue_MyKernelName(…)

‘Use CLU generated functions to save time and make low stress when programming with OpenCL’
#include "cl_clu.h"
#include "simple.cl.h"

void main()
{
    int* mapped = 0;

    // initialize clu
    cl_int status = cluInitialize(0);

    // create my buffer
    int bufferLength = 1024;
    int bufferSize = sizeof(int)*bufferLength;
    cl_mem buf = clCreateBuffer(CLU_CONTEXT, …);

    // get a function that will enqueue my kernel
    clug_Simple s = clugCreate_Simple(0);

    // enqueue my kernel
    clu_enqueue_params params = {0};
    params.nd_range = CLU_ND1(bufferLength);
    status = clugEnqueue_Simple(s, &params, 42, buf);

    // check the results
    mapped = (int*)clEnqueueMapBuffer(clu_DEFAULT_Q, …);
    clEnqueueUnmapMemObject(clu_DEFAULT_Q, …);

    // done using clu
    cluRelease();
}
Using Generator

• Add 1-line custom build step:
  - clu_generator simple.cl -o simple.cl.h -i c:\myproj\inc

- h Show help
- i, -I arg Additional include directories
- n, -N By default, line numbers are prepended to the lines of stringified code. This disables the feature.
- o, -O arg Name of output file (default: appends .h to input file name)
- q, -Q “quiet” output
Summary

• Today, OpenCL development gets a little bit easier
• CLU allows you to write your first OpenCL application in under one hour
• CLU is an open source library developed by a cross industry collaboration
• CLU will run across many vendors platforms today
• Download CLU today at:
Intel OpenCL
The OpenCL advantage:
Maximize the compute power of **all** resources on the platform!

Task-parallel or irregular workloads best suited to an Intel CPU

- Complex game & graphics engine graphs, variable bitrate compression, ...

Highly data-parallel tasks best suited to Intel® HD Graphics

- Film & image post-processing pipelines, graphics, decoding, ...

Today, with 3rd Generation Intel® Core™ Processors with Intel® HD Graphics:
Compute capabilities of CPU and GPU together in one application
DEMO
OpenCL / GL Sharing with CLU
on Ivybridge Ultrabook®!
Acknowledgements

• The entire Visual Computing Advanced Dev Team for contributions and feedback to early versions
• Industry Collaborators at AMD (Ben Gaster, Lee Howes) and other vendors
• Stephen Junkins, Deepti Joshi, and Tim Mattson for strong and continued support of the refinement of this into a deployed industry enabler
• Barry Blount for helping to resolve Legal Issues
References

• [Hux 2012] Internal Presentation on Runtime and Code Generator Overview, 2011.

Backup
Getting Started with CLU: Runtime and CodeGen

Runtime
- Initialization / Shutdown
- ~500 lines of code → ~2 lines of code!
- More time on kernel code, less time on host side mgmt.

Code Generator
- Off-line generator uses .cl files -> .cl.h
  - Produces a C-compatible header containing wrappers for each kernel call scheduled from the host
- Use this as a custom build step during app development
- Less debugging when setting kernel arguments
- Type checking!

Decrease bugs and increase productivity with the CLU API!
Intel® SDK for OpenCL* Applications 2012

A Comprehensive Development Environment for OpenCL Applications

SDK Package

Development Environment
• Supports Both CPU and Intel HD Graphics

Development Tools
• Offline Compiler
• CPU Kernel Debugger

Documentation and Samples
• Getting Started guide
• Optimization Guide

Drivers and Support

Runtime/Drivers
• Support for both CPU and Graphics

Online Resources
• Sample code
• Support Forum
• Tech Articles, forums, and more.

Optimization Tools
• Intel® VTune Amplifier XE
• Intel® Graphics Performance Analyzers (Intel® GPA)
Using Generator

• Add 1-line custom build step:

  - `clu_generator simple.cl -o simple.cl.h -i c:\myproj\inc`

  ![Diagram](input) ![Diagram](output) ![Diagram](Include files)
What’s new in OpenCL 1.2

- Standard released in Q4 2011
- Backward compatible upgrade to OpenCL 1.1
- With a few new additions:
  - **Compile** and **Link** multiple source files into a single shared program
  - Think of it like gcc
  - Create **libraries** and middleware
    - Important for modular & complex code bases, improved re-use & sharing
  - **Surface sharing** with OpenGL, DX9, DXVA, DX10, DX11

Today’s released Beta SDK for OpenCL includes many **preview** OpenCL 1.2 features
Favorite CLU functions

1. cluInitialize – starts system, creates default context and queues for any devices in the platform. Optionally specify the vendor, set custom properties, or provide your own context
2. cluRelease – Release resources created by CLU
3. cluBuildSourceFromFile – Give the runtime a path to a file, CLU will build it and return a CLU program object!
4. cluEnqueue – Use the CLU generated function to enqueue the kernel and its arguments in a single API call!
CLU generator known limitations

- Not a full pre-processor, C grammar, etc.
  - Need to use kernel or __kernel keyword for your kernels...will miss it if you use #DEFINE KERNEL kernel, for example
- Not okay:
  - kernel MYDEFINE void MYDEFINE kernelName
- Okay:
  - kernel MYDEFINE kernelName
- All #includes are expanded, even in comments
- Wrapper ignores preprocessor directives (kernels within #include and within #if/#endif still result in generated code)
What was announced today?

- Intel® SDK for OpenCL* Applications 2013 Beta
- What’s new?
  - Windows 8* support
  - Extended set of optimization and debugging tools
  - New preview CPU OpenCL device
    - Support for OpenCL 1.2 core features and extensions (not conformant)
    - Mixed platform mode with the production Intel® HD Graphics

Prepare for next generation OpenCL products by Intel with
Intel SDK for OpenCL Applications 2013 Beta
## New Platform Programming Tools in Intel® SDK for OpenCL® Applications 2013 Beta

<table>
<thead>
<tr>
<th>Tool Name</th>
<th>Used for</th>
<th>Included with the SDK for OpenCL</th>
<th>Supported devices and operating systems:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Offline Compiler (Standalone)</td>
<td>Create and Build</td>
<td><img src="checkmark.png" alt="checkmark" /></td>
<td><img src="checkmark.png" alt="checkmark" /> <img src="checkmark.png" alt="checkmark" /> <img src="checkmark.png" alt="checkmark" /></td>
</tr>
<tr>
<td>Kernel Builder (Standalone)</td>
<td>Create and Build</td>
<td><img src="checkmark.png" alt="checkmark" /></td>
<td><img src="checkmark.png" alt="checkmark" /> <img src="checkmark.png" alt="checkmark" /> <img src="checkmark.png" alt="checkmark" /></td>
</tr>
<tr>
<td>Offline Compiler (Microsoft Visual Studio® plug-in)</td>
<td>Create and Build</td>
<td><img src="checkmark.png" alt="checkmark" /> <img src="new.png" alt="new" /></td>
<td><img src="checkmark.png" alt="checkmark" /> <img src="new.png" alt="new" /> <img src="new.png" alt="new" /> <img src="new.png" alt="new" /></td>
</tr>
<tr>
<td>Step by Step Kernel Debugger</td>
<td>Debug</td>
<td><img src="checkmark.png" alt="checkmark" /></td>
<td><img src="checkmark.png" alt="checkmark" /> <img src="new.png" alt="new" /> <img src="checkmark.png" alt="checkmark" /></td>
</tr>
<tr>
<td>Intel® VTune™ Amplifier XE</td>
<td>Tune</td>
<td>For purchase on intel.com</td>
<td><img src="checkmark.png" alt="checkmark" /> <img src="new.png" alt="new" /></td>
</tr>
<tr>
<td>Intel® GPA System Analyzers</td>
<td>Tune</td>
<td>For free download on intel.com</td>
<td><img src="checkmark.png" alt="checkmark" /> <img src="new.png" alt="new" /></td>
</tr>
<tr>
<td>Intel® GPA Platform Analyzers with OpenCL Traces Views</td>
<td>Tune</td>
<td>For free download on intel.com</td>
<td><img src="checkmark.png" alt="checkmark" /> <img src="new.png" alt="new" /></td>
</tr>
</tbody>
</table>


Intel® Graphics Performance Analyzers 2012 (Intel® GPA) is available as a free download at [www.intel.com/software/gpa](http://www.intel.com/software/gpa)