IWOCL 2017 - all the talks

An overview of all the tutorials and talks for easy reading.

You can also download the PDF.

Heterogeneous Computing Using Modern C++ with OpenCL Devices - Rod Burns and Ruyman Reyes (Codeplay)

This hands-on session will provide an opportunity to get experience with SYCL using ComputeCpp? Community Edition, a free to use implementation of the SYCL 1.2 standard. Attendees will be shown how to set up ComputeCpp and use it to write their own SYCL code to run on supported GPUs and CPUs.

SYCL is already able to dispatch to heterogeneous devices and it implements C++17 ParallelSTL, augmenting it with ability to dispatch to GPUs in addition to CPUs. This tutorial will demonstrate how to write parallel SYCL code and how to use the Khronos Group's experimental Parallel STL implementation. The course outline is as follows

- Start with a basic SYCL program that shows how to submit queues in a single task and stream-like object, comparing CPU, SYCL and OpenCL versions
- Demonstrate how to access data across host and GPUs using buffers and accessors, the importance of life-time, and basic parallel constructs

Attendees are expected to have programming experience with C++ and a laptop either running Linux or having a VM manager installed such as VirtualBox. The required software will be provided on USB-sticks. This course is suitable for beginners, but is focused on intermediate to advanced parallel programming using C++.

Harnessing the Power of FPGAs with the Intel FPGA SDK for OpenCL- Byron Sinclair, Andrew Ling and Genady Paikin (Intel)

In this tutorial, we will introduce you to the reconfigurable hardware architecture and programming of Field Programmable Gate Arrays (FPGAs).

You will learn why FPGAs have become so popular in recent years, and understand the many advantages of using FPGAs in your HPC application. In particular, we will cover architectural features of FPGAs that make them well suited to many complex operations, including matrix multiplications and convolutions. In addition, we will introduce you to programming FPGAs using the Intel® FPGA SDK for OpenCL, and how specific OpenCL coding techniques can lead to efficient circuits implemented on the FPGA.

Finally, we will go over several case studies where FPGAs have shown very competitive performance when programmed using OpenCL, including convolutional neural nets, FFTs, and astronomy de-dispersion algorithms.

Unlock Intel GPUs for High Performance Compute, Media and Computer Vision Capabilities with Intel OpenCL Extensions - Jeff Mcallister, Biju George, Adam Herr and Ben Ashbaugh (Intel)

The keys to unlock the full performance potential of Intel GPUs for emerging workloads in general compute, media, computer

vision, and machine learning are in the rich suite of Intel OpenCL extensions. These give developers direct access to unique Intel hardware capabilities, which until now have been difficult to master.
This tutorial builds step by step with multiple examples, including:

- How to write high performance general compute applications based on the core concept of OpenCL subgroups.
- How to use additional subgroup operations described in the Intel subgroups and media block read/write extensions.
- Then using the framework of subgroups, we explain the device-side motion estimation extension which leverages the unique Intel GPU media sampler to accelerate motion estimation operations from OpenCL kernels.
- Finally we explain the Video Enhancement (VEBOX) extension, which is an OpenCL host level API extension to leverage a powerful media fixed function unit to accelerate many frame level video enchancement operations.

Faster, smarter computer vision with AI and OpenCL - Uri Levy and Jeffrey Mcallister (Intel)

Learn how to use Intel machine learning and computer vision tools to get from concept to market faster for machine learning applications based on OpenCL and OpenVX. Build two example scenarios: autonomous driving with FPGA inference and a smart camera app using Intel Graphics inference. This presentation will show how a unified set of tools can reduce the complexity of developing heterogeneous machine learning apps ? from training a model with input images, to creating a custom classifier, to building an optimized traditional computer vision pipeline around the classifier to create a full computer vision application

GPGPU Acceleration using OpenCL for a Spotlight SAR Simulator - Eric Balster, Jon Skeans and David Fan (University of Dayton) Marc Hoffman (US Air Force Research Laboratory)

In this paper, OpenCL is used to target a general purpose graphics processing unit (GPGPU) for acceleration of 2 modules used in a synthetic aperture radar (SAR) simulator. Two of the most computationally complex modules, the Generate Return and Back Projection modules, are targeted to an AMD FirePro M5100 GPGPU. The resulting speedup is 2.5X over multi-threaded C++ implementations of those algorithms running on an 8-core Intel I7 2.8GHz processor, 5X over singlethreaded C++ implementations, and 24X over native MATLAB implementations, on average.

Near Real-Time Risk Simulation of Complex Portfolios on Heterogeneous Computing Systems with OpenCL - Javier Alejandro Varela and Norbert Wehn (University of Kaiserslautern)

In this work, we exploit OpenCL to efficiently map the nested simulation of complex portfolios with multiple algorithms on heterogeneous computing systems. Code portability and customizations allow us to profile the kernels on different accelerating platforms, such as CPU, Intel's Xeon Phi and GPU. The combination of OpenCL, a new bit-accurate algorithmic optimization and the extension of an existing numerical interpolation scheme allows us to achieve 1000x speedup compared to the state-of-the-art approach. Our system design minimizes costly host-device transfers and global memory, enabling complex portfolios to be easily scaled.

A Performance and Energy Evaluation of OpenCL-accelerated Molecular Docking - Leonardo Solis Vasquez and Andreas Koch (Technische Universität Darmstadt)

This work presents an OpenCL implementation of AutoDock, and a corresponding performance evaluation on two different platforms based on multi-core CPU and GPU accelerators. It shows that OpenCL allows highly efficient docking simulations, achieving speedups of ?4x and ?56x over the original serial AutoDock version, as well as energy efficiency gains of ?2x and ?6x. respectively. To the best of our knowledge, this work is the first one also considering the energy efficiency of molecular docking programs.

Assessing the feasibility of OpenCL CPU implementations for agent-based simulations - Nuno Fachada and Agostinho Rosa (Instituto Superior Técnico, Portugal)

In this paper we evaluate the feasibility of using CPU-oriented OpenCL for high-performance simulations of agent-based models. We compare a CPU-oriented OpenCL implementation of a reference ABM against a parallel Java version of the same model. We show that there are considerable gains in using CPU-based OpenCL for developing and implementing ABMs, with speedups up to 10x over the parallel Java version on a 10-core hyper-threaded CPU.

Enabling FPGAs as a True Device in the OpenCL Standard - Vincent Mirian and Paul Chow (University Of Toronto)

As FPGA capacities continue to increase, the ability to partition and partially reconfigure the FPGA will become even more desirable. The fundamental issue is how FPGAs are currently viewed as devices in the OpenCL model. In this paper, we propose a small change to the OpenCL definition of a device that unlocks the full potential of FPGAs to the programmer.

Applying Models of Computation to OpenCL Pipes for FPGA Computing - Nachiket Kapre and Hiren Patel (University of Waterloo)

We propose imposing a communication discipline inspired from models of computation (e.g.Ptolemy) such as SDF (synchronous dataflow), bulk synchronous (BSP), or Discrete Event (DE). These models offer a restricted subset of communication patterns that enable implementation tradeoffs and deliver performance and resource guarantees. This is useful for OpenCL developers operating within the constraints of the FPGA device. We hope to facilitate a preliminary analysis and evaluation of supporting these patterns in OpenCL and quantifying associated FPGA implementation costs.

Accelerating Applications at Cloud Scale using FPGAs - Sarah Siripoke, Fernando Martinez Vallina and Spenser Gilliland (Xilinx)

The acceptance and success of cloud computing has given application developers access to computing and new customers at a scale never seen below. The inherent ability of an FPGA to reconfigure and be workload optimized is a great advantage given the fast-moving needs of cloud computing applications. In this talk we will discuss how users can develop, accelerate and deploy accelerated applications in the cloud at scale. You will learn how to get started on a turn-key OpenCL development environment in the cloud using Xilinx FPGAs.

Creating High Performance Applications with Intel's FPGA OpenCL SDK - Andrew Ling, Utku Aydonat, Davor Capalija, Shane O'Connell and Gordon Chiu (Intel)

After decades of research, High-Level Synthesis has finally caught on as a mainstream design technique for FPGAs. However, achieving performance results that are comparable to designing at a hardware description level still remains a challenge. In this talk, we illustrate how we achieve world class performance results on HPC applications by using OpenCL. Specifically, we show how we achieve 1Tflop of performance on a matrix multiply and over 1.3Tflops on a CNN application, run on Intel's 20nm Arria 10 FPGA device. Finally, we will describe spatial coding techniques that lead to efficient structures, such as systolic-arrays, to ensure that the FPGA runs efficiently.

Symphony - Task Scheduling and Memory Management in Heterogeneous Computing - Amit Jindal and Wenjia Ruan (Qualcomm Technologies)

Task scheduling and memory management are challenges that make Heterogeneous Computing difficult for the masses. There are several programming models and tools that exist targeting partitioning of workload and accessibility of data between CPU and GPU. We have developed and deployed Symphony SDK ? a framework that makes workload partitioning, scheduling and memory management ?simple' for developers. In this talk, we will introduce Symphony architecture, elaborate how existing OpenCL kernels can be reused with heterogeneous task synchronization, task scheduling, and memory management capabilities of Symphony. We will also share real-world cases where Symphony has provided 2x-6x performance speed-ups.

CUDA-on-CL: A compiler and runtime for running modern CUDA c++11 applications on OpenCL 1.2 devices - Hugh Perkins (ASAPP)

Cuda-on-cl addresses the problem of creating and maintaining OpenCL forks by leaving the reference implementation entirely in NVIDIA CUDA, and writing both a compiler and a runtime component, so that any CUDA c++11 application can in theory be

compiled and run directly on any OpenCL 1.2 device. We use Tensorflow framework as a case-study, and demonstrate the ability to run Tensorflow and Eigen kernels directly, with no modification to the original CUDA source-code. Performance studies are also undertaken, and show that the cuda-on-cl program runs at about 25% of the original CUDA-compiled version.

OpenCL in Scientific High Performance Computing?The Good, the Bad, and the Ugly - Matthias Noack (Zuse Institute Berlin)

We present experiences with utilising OpenCL alongside C ++ , MPI, and CMake in two real-world scientific codes. Our targets are a Cray XC40 supercomputer with multi- and many-core (Xeon Phi) CPUs, as well as multiple smaller systems with Nvidia and AMD GPUs. We shed light on practical issues arising in such a scenario, like the interaction between OpenCL and MPI, discuss solutions, and point out current limitations of OpenCL in the domain of scientific HPC from an application developer's and user's point of view.

Accelerated Machine Learning Using TensorFlow and SYCL on OpenCL Devices - Andrew Richards, Mehdi Goli and Luke Iwanski (Codeplay)

Codeplay has been working with Google to add SYCL back-end support in TensorFlow, one of the most popular machine learning frameworks, enabling developers to use OpenCL devices with their machine learning applications. SYCL provides an abstraction layer that simplifies parallel development, giving developers access to the computing power of OpenCL devices and reducing the amount of code required. Andrew Richards will talk about how machine learning applications can harness the power of OpenCL using open standards and how, by using SYCL, TensorFlow can be extended to include customized operations running on OpenCL devices.

Analyzing and improving performance portability of OpenCL applications via auto-tuning - James Price and Simon McIntosh-Smith (University of Bristol)

In this talk, we present an approach for analyzing performance portability that exploits that black-box nature of automatic performance tuning techniques. We demonstrate this approach across a diverse range of GPU and CPU architectures for two simple OpenCL applications. We then discuss the potential for auto-tuning to aid the generation of performance portable OpenCL kernels by incorporating multi-objective optimization techniques into the tuning process.

Wavefront Parallel Processing on GPUs with an Application to Video Encoding Algorithms - Biju George and Ben Ashbaugh (Intel)

In this presentation we focus on the application of the wavefront pattern to design efficient GPGPU implementations of video encoding algorithms using OpenCL kernels. We present our experiences in implementing and evaluating four solutions of WPP for inter and intra estimation for AVC on GPUs. We explain the reasoning behind each solution and present the results of our analysis.

Challenges and Opportunities in Native GPU Debugging with OpenCL - Uri Levy (Intel)

In this technical session we'll present the open architectural design of the debugger and how it fits into the OpenCL JIT compilation flow and the underlying compute technology of the HW with focus on Intel processor graphics. We'll demonstrate a show case on how to natively work with the debugger to solve functional bugs, as-well-as low-level debugging techniques on SIMD thread level which help to solve complex issues such as misaligned or out of range accesses to localglobal memory, stack overflows, Illegal instructions, etc. Finally, we'll cover the challenges in debugging

Modeling Explicit SIMD Programming with Subgroup Functions - Biju George and Ben Ashbaugh (Intel)

In this presentation, based on our experience in developing publicly released vendor extensions based on subgroups, we explain the advantages of the ?explicit SIMD? programming paradigm using OpenCL subgroup and how the subgroups framework can be leveraged to: (1) Model features for performance in OpenCL that are commonly available in programming languages or interfaces based on an ?explicit SIMD? programming paradigm such as the AVX intrinsics supported in GCC; and to (2) Model features to expose functionality available in GPU accelerator units that are more conveniently and efficiently exposed using a block API.