L9 - OpenCL

Parallel programming with OpenCL

Objectives

- Learn parallel programming with OpenCL.
- Know what (not) to expect from parallel programming.
- Understand heavy multithreading and how it is mapped to the hardware.
- Measure OpenCL code performance, locate and solve bottlenecks.
- Write efficient OpenCL code.

Depending on the hardware environment, exercises will be run on either multi-core CPUs, nVidia or ATI GPUs.

Course environment

- One PC under Windows for two trainees, with either
  - Intel OpenCL SDK (needs a recent CPU, core i3 or better, and Windows 7)
  - nVidia SDK (needs a recent workstation-class nVidia graphic interface)
  - ATI SDK (needs a recent workstation-class ATI graphic interface)

Exercise: For on-site training sessions, contact us to check the needed configuration for PC used during hands-on labs.

Pre-requisites

- Good knowledge of the C language

Target Audience

- Any embedded systems engineer or technician with the above prerequisites.

Course Outline

First day

Introduction to OpenCL

- History
  - OpenCL 1.2
  - OpenCL 2.2
  - OpenCP/EP (Embedded Profile)
- Design goals of OpenCL
  - CPUs, GPUs and GPGPUs
  - Data-parallel and Task-parallel
  - Hardware related and portable
- Terminology
  - Host / Device
  - Memory model
  - Execution Model

The OpenCL Architecture

- The OpenCL Architecture

    - Platform Model
    - Execution Model
    - Memory Model
    - Programming Model
- The OpenCL Software Stack
- Example

Exercise: Installation and test of the OpenCL SDK

The OpenCL Host API

- Platform layer
    - Querying and selecting devices
    - Managing compute devices
    - Managing computing contexts and queues
    - The host objects: program, kernel, buffer, image

Exercise: Write a platform discovery and analysis program (displaying CPUs, GPUs, versions...)

- Runtime
    - Managing resources
    - Managing memory domains
    - Executing compute kernels

Exercise: Write an image loader program, transferring image to/from compute devices

- Compiler
    - The OpenCL C programming language
    - Online compilation
    - Offline compilation

Second day

The Basic OpenCL Execution Model

- How code is executed on hardware
    - Compute kernel
    - Compute program
    - Application queues
- OpenCL Data-parallel execution
    - N-dimensional computation domains
    - Work-items and work-groups
    - Synchronization and communication in a work-group
    - Mapping global work size to work-groups
    - Parallel execution of work-groups

Exercise: Compile and execute a program to square an array on the platform computing nodes

The OpenCL Programming Language

- Restrictions from C99
- Data types
    - Scalar
    - Vector
    - Structs and pointers
    - Type-conversion functions
    - Image types

Exercise: Rewrite the square program to use vector operations

- Required built-in functions
    - Work-item functions
    - Math and relational
    - Input/output
    - Geometric functions

    - Synchronization
- Optional features
    - Atomics
    - Rounding modes

Exercise: Write and execute an image manipulation program (Blur filter)

Third day

Advanced OpenCL Execution modes

- Profiling

Exercise: Enhance the image manipulation program to measure kernel computation time

- The OpenCL Memory Model
    - Global Memory
    - Local Memory
    - Private Memory
- OpenCL Task-parallel execution
    - Optional OpenCL feature
    - Native work-items

Exercise: Simulate the N-Body problem, displaying data using OpenGL

Efficient OpenCL

- When (not) to use OpenCL
- Code design guidelines
- Explicit vectorization

Exercise: Explore vectorisation on an image rotation kernel

- Memory latency and access patterns
    - ALU latency
    - Using local memory

Exercise: Enhance the Blur filter program to investigate memory optimisations

- Synchronizing threads
- Warps/Wavefronts, work groups, and GPU cores