# oL9 - 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.

Exercises will be run on multi-core CPUs with nVidia GPU running under Linux.

## Pre-requisites

- Good knowledge of the C language

## Course environment

- Theoretical course
  - PDF course material (in English)
  - Course dispensed using the Teams video-conferencing system
  - The trainer to answer trainees questions during the training and provide technical and pedagogical assistance through the Teams video-conferencing system
- Practical activities
  - Practical activities represent from 40% to 50% of course duration
  - One Online Linux PC per trainee for the practical activities
  - The trainer has access to trainees Online PCs for technical and pedagogical assistance.
  - Example code, labs and solutions
- One PC under Linux per trainee, with a multi-core CPU and an nVidia GPU

## Duration

- Total: 20 hours
- 4 sessions, 5 hours each (excluding break time)
- From 40% to 50% of training time is devoted to practical activities
- Some Labs may be completed between sessions and are checked by the trainer on the next session

## Target Audience

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

# Course Outline

## First Session

### 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

## Second Session

### 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

### 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

# Third Session

## 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)

# Fourth Session

## 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