**OpenCL, assignment 3**

# Introduction

In Assignment 2, we used a program (CLGetPlatforms) to find out which OpenCL platform you had to use on your system. For finding more information about that platform, we could make another, more elaborate program, but luckily there are freeware programs available that help us out here.

One of these is GPU Caps Viewer, which is very useful to get all information about your OpenCL platforms. Install this program from <http://ozone3d.net/gpu_caps_viewer/>, and run it.   
Make a note of the following information:

* Global memory size.
* Local memory size.
* Work Item Sizes (the maximal size of a Workgroup for each separate dimension).
* Work group size (the maximal total number of threads in a Workgroup)

The main goal of the following assignment is to learn to apply synchronization (on the host and on the GPU), and how to allocate local memory.

# About the reduction algorithm

The task for this week is to fully implement the kernels mentioned in the slides of this week. You might want to use the program of last week (Mandelbrot) as starting point.   
Use again the functions checkError and build\_program from opencl\_utils.

Take care of the following specifics:

* Implement kernel 1 and kernel 2 from the slides.   
  Use dynamically allocated Local Memory, and barriers in the right places.  
  Optional: implement also the last kernel (kernel 3).
* Specify the length of the array, the size of the Workgroups and the kernel to use either as hard-coded constants in the program, as command line arguments, or by user input.
* Use an array that is really big (several MB; check the Global Memory size noted above to see how far you can go).
* As described in the slides, this program does not use just one singe kernel call, but it calls the same kernel several times to reduce the whole array into one single number. How often the kernel must be called depends on the length of the array and on the size of the Workgroups.
* Your program should print at least the following information:
  + The length of the array used.
  + The time needed to copy the array from the host to the device.
  + The total time that is used by all the kernels calls together.
  + The type of kernel used (according to the slides).
  + The sum that was calculated by the kernel.
  + Of course more information might be useful for debugging purposes.
* For each kernel, try different workgroup sizes to see which is most efficient (check the maximal Workgroup size noted above). Summarize your results in a table where we can see the execution times for the two kernels and for different Workgroup sizes.