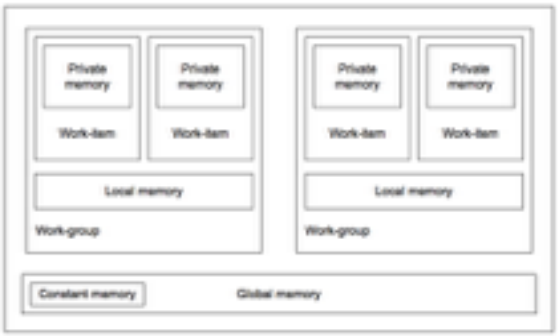
OpenCl & OpenCV Info

OpenCl: OpenCl is a set of open source libraries designed to be able to run on any type of device with an OpenCL platform such as CPU, GPU, DSP, etc. To run something on a GPU, you must create OpenCL kernels. These kernels are loaded, compiled, and run by a C++ program.

Normally, To keep the CPU running the C++ program from accessing and modifying data the kernel is using, you must copy the buffers holding data from CPU memory to GPU memory and vice versa when the kernel finishes execution. On the CPU side, this can be done with a memcpy or telling the functions that create buffers for data that you want a new pointer to the GPU dataset when it completes. In the GPU, you create local variables to store data and then ensure data is synced by calling ***barrier(CLK_LOCAL_MEM_FENCE)*** after local variables are created and before processing any data. An example of this might look like:



GPU memory structure

```
void kernel some_kernel(__global DATATYPE src,
                        __global DATATYPE dst,
                        size_t value)
{
    //Define local variables
    …
    //Populate variables with data from parameters
    …
    barrier(CLK_LOCAL_MEM_FENCE);

    //Can now process data since its synced and no memory
    //will be corrupted by operating on it
    …
}
```

However, the Mali-T628 GPU on the ODroid XU3 is designed slightly different. Instead of having to copy memory from the CPU, the Mali-T628 shares its memory with the CPU. So there is no copying of data from CPU to GPU and no need to use the barrier() function unless you need synchronization within the GPU kernel. Thus, you can just use the parameters given and operate directly on those instead of forcing exchanges to local variables. You do have to be a bit more careful in ensuring data is only operated on by one process though. Some of the differences include:

- Unified memory system
- Local and private memory is physically global memory
- Moving data from global to local or private memory typically does not improve performance
- Copying of data is not required

A really good reference to how the Mali GPU works differently what is explained above is here:
http://infocenter.arm.com/help/topic/com.arm.doc.dui0538f/DUI0538F_mali_t600_opencl_dg.pdf

Examples of OpenCL on the Mali GPU here:
http://malideveloper.arm.com/downloads/deved/tutorial/SDK/opencl/index.html

Reference for OpenCL here:
https://www.khronos.org/registry/cl/sdk/1.1/docs/man/xhtml/

OpenCV: This is a powerful set of libraries to process almost all types of data. Anything from images to audio, AI classifiers to complex math can be used in OpenCV. It has many, many different options for image processing and even support for Nvidia CUDA enabled GPUs. The basic data type in OpenCV is the Mat, or matrix. It is essentially a large array that has some header info and then can hold almost any type of data. The functions and operations were designed to run on a CPU, not a GPU, so they are executed sequentially, not in parallel. Thus, when processing images, only one pixel is operate on at a time. This is why there is support for parallel libraries, because parallelism not intrinsic to OpenCV like it is in OpenCL.

There is support for many types of inputs and because it is written in C++, it is easy to include data processing functions in your own code without much effort. The only tricky part is getting OpenCV working in the device. The libraries must be compiled into binaries. This is done through a tool called CMake. Once you download the OpenCV libraries, you use CMake to compile them into a single folder. After that, it is useable. These are some of the links to get started installing and building OpenCV:

http://opencv.org/
http://docs.opencv.org/
http://www.cmake.org/cmake-tutorial/

The tutorials page of OpenCV in the second link is very useful showing all the things you can use it for as well as showcasing many of the different ways to achieve the same result.

…And, as always, one of the best resources for both OpenCL and OpenCV and anything else :)
http://stackoverflow.com/