

LAB 6

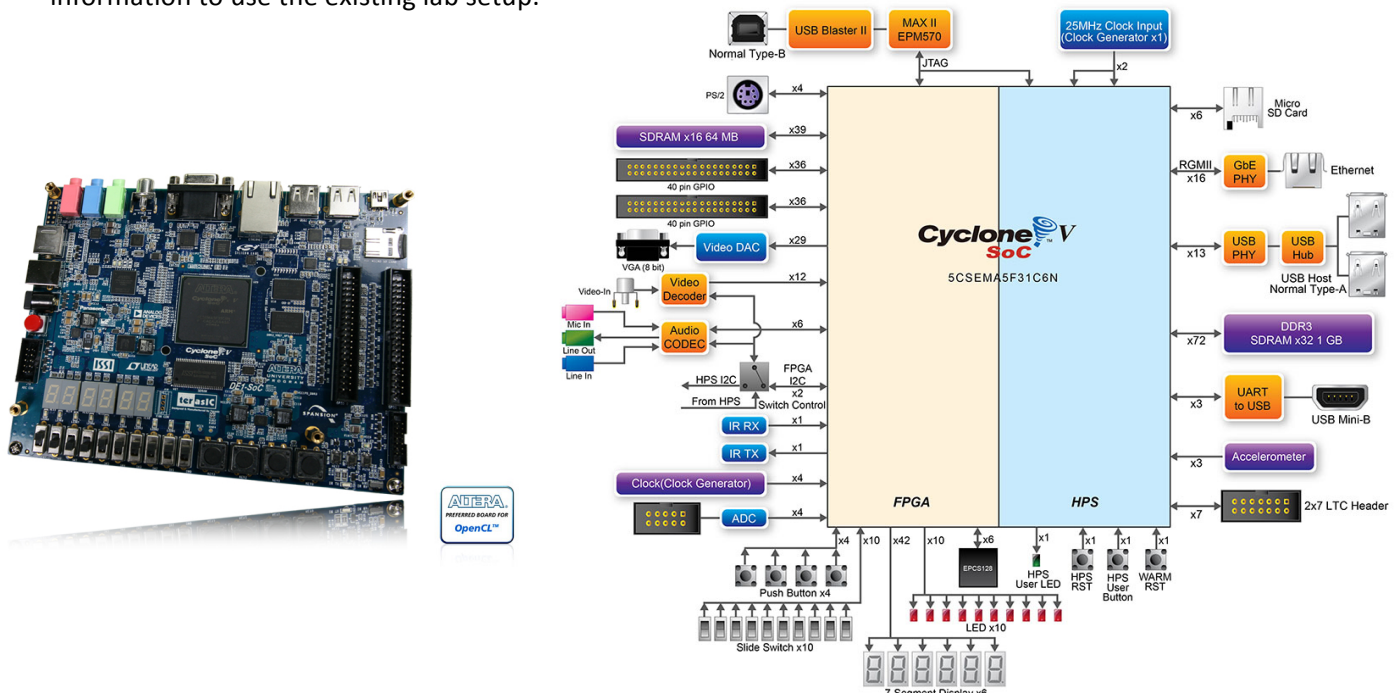
– OpenCL on FPGA –

Heterogeneous High Performance Computing (2021/2022)
Master in Electrical and Computer Engineering

1. Introduction and initial tests to the board and host development system

The computers in the lab already have Altera Quartus II 14.1 with Altera OpenCL SDK and board support package (BSP) for the DE1-SoC card. To install on another linux computer the necessary files and how-to full instructions are at: http://home.deec.uc.pt/~jlobo/altera/OpenCL/_OpenCL_Q14_linux, for windows check the folder https://home.deec.uc.pt/~jlobo/altera/OpenCL/_OpenCL_Q14_win, for further details and distributions you should go to the Intel FPGA or Terasic website.

In the inforestudante online university system you will find more detailed information about running DE1-SoC Linux and using the Altera OpenCL SDK, but in this lab assignment you will find the essential information to use the existing lab setup.



The base directory is `/usr/local/altera/14.1/hld/`, and the BSP files are in `/usr/local/altera/14.1/hld/board/de1soc/`. Among them are the examples. In this setup you have a lab computer for development, you connect to the DE1-SoC board via USB to have a terminal. Once setup, you can use an Ethernet cable to transfer files. On the board you have the host system running linux on the ARM processor (HPS), and the FPGA.

But let's first start the board and see how to work with it.

1.1 Connect and test the board

The DE1-SoC card is already properly configured and has an SD card that boots into linux and has support for the OpenCL runtime SDK. To gain access to the system you must connect an USB cable to the host computer and use a terminal window as follows:

connect host via usb:

```
dmesg | grep tty
```

check what usb tty you have, and use it

```
sudo screen /dev/ttyUSB0 115200
```

another alternative is to use gtkterm:

```
sudo gtkterm -p /dev/ttyUSB0 -b 115200 &  
(if not installed: sudo apt-get install gtkterm )
```

You can now power up the board, the terminal should show the linux boot starting, just wait for the login prompt. To login just enter as root, without password.

After logging in, you need to start the Altera Runtime Environment for OpenCL :

```
source ./init_opencl.sh
```

And, to check that the Altera Runtime Environment for OpenCL is setup, try the following:

```
aocl version
```

At any time you can also run a diagnose that checks that the kernel driver is installed

```
aocl diagnose
```

You can use normal unix commands, such as **ls -la** to see the contents of the folder, or **pwd** to know which file system folder it is in, tab to fill in names, etc.

To test a precompiled example, change to the vector_add folder by doing:

```
cd vector_add
```

and then configure the FPGA with the file that resulted from a previous compilation and synthesis of the vectorAdd.cl code.

```
aocl program /dev /ac10 vectorAdd.aocx
```

The above loads the FPGA side of the SoC chip with the full configuration, you can now run the corresponding program on the ARM side of the SoC, where the linux is running, and test the program

```
./vectorAdd
```

If all goes well, it should indicate that it has passed, and the processing time. This confirms that you have the entire system working correctly. Leave this terminal window connected to the board for later use. You can now go back to the computer and compile and perform synthesis for a full example from scratch.

1.2 Compile and perform synthesis for a full example

For the compilation and synthesis of a full example from scratch, we start with a computer terminal window and start out by enforcing the correct configuration of the environment:

```
source /usr/local/altera/14.1/hld/init_openccl.sh
```

(if need be: source /etc/environment)

And check with:

```
aocl version  
aoc --list-boards
```

The following instructions indicate the examples folder from the DE1-SoC board support package and Altera OpenCL SDK, but it is best that you copy the full examples folder to some sandbox folder in your home directory. To test the compilation and synthesis, we start with the FPGA side to generate a .aocx file:

```
cd /usr/local/altera/14.1/hld/board/de1soc/examples/vector_add/  
aoc device/vectorAdd.cl --profile -v --report --sw-dimm-partition -o bin/vectorAdd.aocx
```

A full synthesis on Quartus is done for the design from the OpenCL compilation. An estimate of used resources is shown, followed by the synthesis. This process takes a while, but in the end gives the file needed to have the kernel and all OpenCL setup in the FPGA. You can see in the generated folders the corresponding Quartus project and see the utilization rate, although netlist view does not help much to understand what is implemented. The above code should take about 12 minutes.

In parallel you can start the compilation for the ARM side of the DE1-SoC board, but a cross-compiler is needed to generate the code for ARM. In Linux you could use gcc, in windows supposedly VisualStudio, but many dependencies and specific libraries are required, but there is something more practical provided by Altera SoC EDS. To compile the code to have the ARM program, you can open another computer terminal window, change to the desired directory, and use a specific terminal shell setup by EDS. Start another terminal window to run a shell for make with EDS.

```
sudo /usr/local/altera/14.1/embedded_command_shell.sh  
source /usr/local/altera/14.1/hld/init_openccl.sh
```

And now go to the desired folder and build the executable file using the makefile:

```
cd /usr/local/altera/14.1/hld/board/de1soc/examples/vector_add/  
make
```

You now have all that is needed to run on the board, **vector_add** and **bin/vectorAdd.aocx**, but the files must be sent to the board.

1.3 Setup network to transfer files to and from DE1-SoC to the computer

You need a means to transfer files from the computer to the DE1-SoC board linux filesystem, the best way is to setup the network connection. Connect an Ethernet cable from the board to a workbench socket. Run **ifconfig** on the computer to check the IP address, at the LSD lab it should be 10.206.106.<PC_number_1-20>, such as 10.206.106.9.

On the DE1-SoC terminal check the network devices configuration, and set to board to the desired address, in this case similar to the computer but add 200:

```
ifconfig
ifconfig eth0 10.206.106.209    (or 10.10.106.<PC_number_1-20 + 200>)
ifconfig
```

From another computer terminal try to ping 10.206.106.209, if it fails, check the cable and IPs with ifconfig.

If a network connection is not available a USB memory stick can also be used to transfer files. You must connect the USB stick with the board powered down, and then at start-up the linux system should recognize the device, but you have to umount and mount each time you swap the card:

```
mkdir/mnt/usbfs (required only the first time)
dmesg
mount /dev /sda1/mnt/usbfs
```

The files on the USB stick are mounted on the file system at /mnt /usbfs . In the end you should do the **umount**, but if you did not write or update any file it is not necessary.

From the computer terminal, change to the desired directory, and use ftp as follows:

```
sftp root@10.206.106.209
put <file_name>
quit
```

The files will be placed at /home/root. You can also *get* file, *cd*, *ls*, etc...

You can leave this terminal window open, i.e. run sftp but do not quit, so that you have the ftp connection active whenever needed.

1.4 Test the compiled example

Getting back to the developed code indicated above, you can now send the files to the board:

```
sftp root@10.206.106.209
put vector_add
put bin/vectorAdd.aocx
quit
```

The files will be placed at /home/root, and you can test them by configuring the FPGA and running the arm code from the DE1-SoC side. As previously explained, using the terminal window connected to the board:

```
aocl program /dev /ac10 vectorAdd.aocx
./vector_add
```

Since in the generation of the vectorAdd.aocx the profile option was chosen (--profile), you can now collect the profile data from the **profile.mon** file, using the **sftp** open window:

```
get profile.mon
```

From the computer terminal used to perform the synthesis, or any other after setting up the Altera OpenCL SDK environment, launch the profile visualisation with profile.mon and corresponding .aocx by running:

```
export QUARTUS_ROOTDIR=/usr/local/altera/14.1/quartus
aocl report vectorAdd.aocx profile.mon
```

You can now explore the folder `/usr/local/altera/14.1/hld/board/de1soc/examples` and develop your own code.

2. Multiply two arrays

Using the above example as a base, implement a program to multiply two arrays, just as you did in lab5 in CUDA.

3. Convert RGB image to grayscale

Using the above example as a base, implement a program to convert an RGB image to grayscale, i.e.:

For each pixel(r g b) at (I,J)

do: $\text{grayPixel}[I,J] = 0.21 * r + 0.71 * g + 0.07 * b$

