FL24 CSE565M - Lab 1

Instructor: Dr. Anthony Cabrera

TOC

Building and Running Simulations

1. Log into build server using either the VNC server or just the terminal directly (without the GUI)

2. Navigate to (or create and navigate to) the directory where you want Lab 1 to live.

   ```
   mkdir -p /home/your_username/path/to/where/you/want/lab1/to/live
   ```

   e.g.,

   ```
   mkdir -p /home/cabrera/lab1
   ```

   Create an environment variable for this path

   ```
   export LAB1PATH=/home/your_username/path/to/where/you/want/lab1/to/live
   ```

   I'll now refer to this project path by the `LAB1PATH`.

3. Clone this project to the repository

   ```
   git clone https://github.com/cabreraam/Vitis_Accel_Examples.git
   ```

4. Run these environment setup commands

   ```
   source /tools/Xilinx/Vitis/2023.1/settings64.sh
   ```

```
source /opt/xilinx/xrt/setup.sh
```

To verify that the scripts ran, you can issue

```
echo $PATH
```

and should observe output similar to this:

```
/opt/xilinx/xrt/bin:/tools/Xilinx/Vitis_HLS/2023.1/bin:/tools/Xilinx/Vitis/2023.1/bin:/tools/Xilinx/Vitis/2023.1/gnu/microblaze/lin/bin:/tools/Xilinx/Vitis/2023.1/gnu/arm/lin/bin:/tools/Xilinx/Vitis/2023.1/gnu/microblaze/linux_toolchain/lin64_le/bin:/tools/Xilinx/Vitis/2023.1/gnu/aarch32/lin/gcc-arm-linux-gnueabi/bin:/tools/Xilinx/Vitis/2023.1/gnu/aarch32/lin/gcc-arm-none-eabi/bin:/tools/Xilinx/Vitis/2023.1/gnu/aarch64/lin/aarch64-linux/bin:/tools/Xilinx/Vitis/2023.1/gnu/aarch64/lin/aarch64-none/bin:/tools/Xilinx/Vitis/2023.1/gnu/armr5/lin/gcc-arm-none-eabi/bin:/tools/Xilinx/Vitis/2023.1/tps/lnx64/cmake-3.3.2/bin:/tools/Xilinx/Vitis/2023.1/aietools/bin:/tools/Xilinx/Vivado/2023.1/bin:/tools/Xilinx/DocNav:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin
```

5. Change into the `VitisAccelHelloWorld/` directory of the cloned project:

```
cd ${LAB1PATH}/Vitis_Accel_Examples/VitisAccelHelloWorld/hello_world
```

6. Issue the following commands to build for SW and HW emulation:

   For SW emulation:

```
cp
${LAB1PATH}/Vitis_Accel_Examples/hello_world/_x.sw_emu.xilinx_u280_gen3x16_xdma_1_202211_1/emconfig.json .

make all TARGET=sw_emu
PLATFORM=/opt/xilinx/platforms/xilinx_u280_gen3x16_xdma_1_202211_1/xilinx_u280_gen3x16_xdma_1_202211_1.xpfm
```

   For HW emulation:

```
make all TARGET=hw_emu
PLATFORM=/opt/xilinx/platforms/xilinx_u280_gen3x16_xdma_1_202211_1/xilinx
_u280_gen3x16_xdma_1_202211_1.xpfm
```

The SW emulation build shouldn't take too long -- ~1 minute from my testing -- but the HW emulation build will take longer -- ~9 minutes in my testing.

7. Run the applications for both SW and HW emulation
    - For SW:

```
${LAB1PATH}/Vitis_Accel_Examples/VitisAccelHelloWorld/hello_world
export XCL_EMULATION_MODE=sw_emu
./hello_world_xrt -x \
./build_dir.sw_emu.xilinx_u280_gen3x16_xdma_1_202211_1/vadd.xclbin
```

Copy and paste your output.

    - For HW:

```
export XCL_EMULATION_MODE=hw_emu
./hello_world_xrt -x
./build_dir.hw_emu.xilinx_u280_gen3x16_xdma_1_202211_1/vadd.xclbin
```

Copy and paste your output.

Examing Hello World Code

Device Code

1. Open `${LAB1PATH}/Vitis_Accel_Examples/VitisAccelHelloWorld/hello_world/src/vadd.cpp` in a text editor of your choice.
2. Instead of adding the two inputs, modify the code so that you subtract the second input from the first input. Describe what you did.

> **NOTE**: don't worry about changing variable names to reflect subtracting vs. adding.

4. Now repeat the SW steps from steps 6 and 7 in the previous section. Record the output. Is it what you expect? Why is it important to check for functional correctness in the SW step vs the HW steps?
5. Modify `${LAB1PATH}/Vitis_Accel_Examples/VitisAccelHelloWorld/hello_world/src/vadd.c`

`pp` to reflect the change you made in step 3 above. Describe what you did.

Building and Running On Actual Hardware

1. Navigate back to the project directory

   ```
   cd ${LAB1PATH}/Vitis_Accel_Examples/VitisAccelHelloWorld/hello_world
   ```

2. Create a `tmux` session

   ```
   tmux new -t build
   ```

3. In this `tmux` session, verify like that the `PATH` variable is set correctly like we did in the previous section. If it is not, you will have to issue the `source` commands from the previous section.

4. Issue the following command:

   ```
   make all TARGET=hw
   PLATFORM=/opt/xilinx/platforms/xilinx_u280_gen3x16_xdma_1_202211_1/xilinx
   _u280_gen3x16_xdma_1_202211_1.xpfm
   ```

   This command should take a while -- it took ~2 hours when I ran it.

   At this point, you can use the keystroke

   ```
   ctrl+b, d
   ```

   to detach from the terminal. This will bring you back to the original terminal session.

   To check if your build is done, you can issue

   ```
   tmux attach
   ```

   in the terminal. Then when you are done checking, you can detach from the terminal again.

5. Once the build is done, we need to move the build over to an OCT Server.

   Copy the xclbin and the executable (`hello_world_xrt`) to the cloudlab server. Before doing this, make sure you place the CloudLab private key in the `~/.ssh` directory on the

NERC machine.

> **NOTE**: If your CloudLab private key was generated using PuTTY, it needs to be converted to OpenSSH format before being placed on the NERC machine. Follow these instructions for key conversion.

For example, if you want to copy these files to the home directory of `pc151.cloudlab.umass.edu`, you use the following command:

```
scp -i ~/.ssh/<your cloudlab private key> hello_world_xrt
./build_dir.hw.xilinx_u280_gen3x16_xdma_1_202211_1/vadd.xclbin <your user
name>@pc151.cloudlab.umass.edu:~
```

On the Cloudlab server run the application.

```
./hello_world_xrt -x vadd.xclbin
```

Record your output.

At this point, you have successfully built and run something on a U280 FPGA!