Exercise Manual

for

Inference with FPGAs

Software Requirements

64-bit Linux* Software Development Environment with g++
Intel® Distribution of OpenVINO™ toolkit Linux for FPGA version R3

Hardware Requirements

Intel® Programmable Acceleration Card with Intel® Arria® 10 FPGA GX

Exercise 2

Perform Inference on an FPGA

In this exercise we will continue with the application from exercise 1. If you haven't yet completed that exercise, you'll need to go back and finish that first.

Step 1. Setup FPGA Lab Environment

____ 1. Open a terminal in your Linux system

____ 2. Go to the <Lab Dir> from Exercise 1

____ 3. Examine the environment script init_openvino.sh

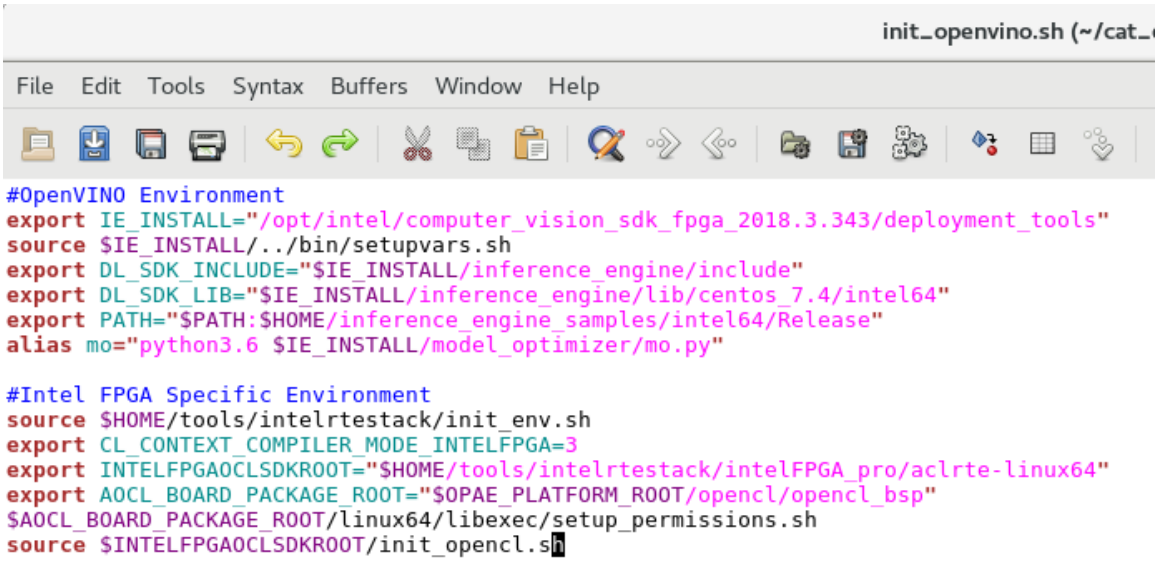


Figure 1. Environment script init_openvino.sh

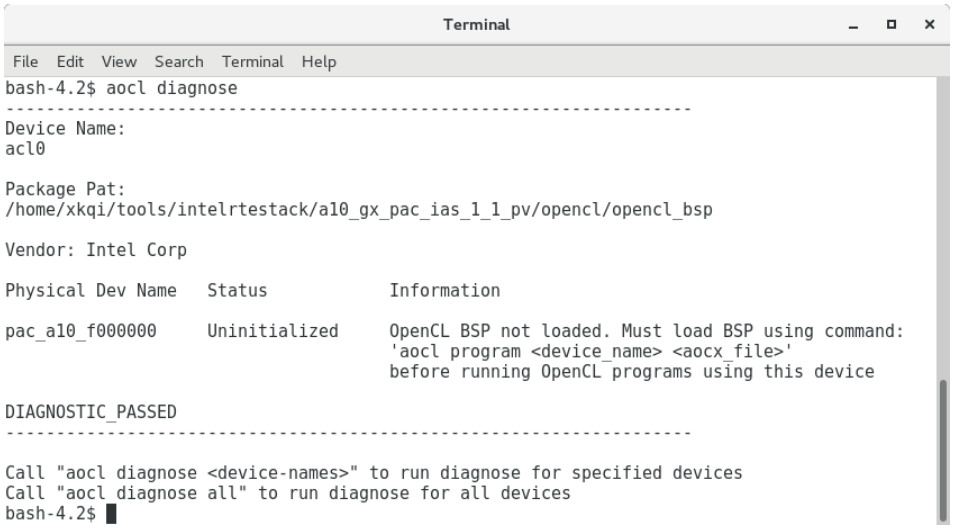
In the script there are two sections, OpenVINO™ toolkit environment and Intel® FPGA envirnoment settings.

____ 4. Edit the script if necessary to make sure all the FPGA paths are correct

____ 5. Source init_opencl.sh from the terminal if you haven't already done so

a. source init_openvino.sh

____ 6. Run "aocl diagnose", you should see an FPGA board connected



```
Terminal
File  Edit  View  Search  Terminal  Help
bash-4.2$ aocl diagnose
-----------------------------------------------------------------------
Device Name:
acl0

Package Pat:
/home/xkqi/tools/intelrtestack/a10_gx_pac_ias_1_1_pv/opencl/opencl_bsp

Vendor: Intel Corp

Physical Dev Name   Status            Information

pac_a10_f000000     Uninitialized     OpenCL BSP not loaded. Must load BSP using command:
                                      'aocl program <device_name> <aocx_file>'
                                      before running OpenCL programs using this device

DIAGNOSTIC_PASSED
-----------------------------------------------------------------------

Call "aocl diagnose <device-names>" to run diagnose for specified devices
Call "aocl diagnose all" to run diagnose for all devices
bash-4.2$
```

Figure 2. Screen capture showing FPGA board is connected

____ 7. Go to the directory where all the DLA FPGA images are located

a. cd $IE_INSTALL/../a10_dcp_bitstreams

If you're not using an Intel® Programmable Acceleration Card with Intel® Arria® 10 GX FPGA, go to the appropriate bitstream directory.

____ 8. Examine the contents of the directory

You should see many bitstreams optimized for different topologies and data types.

____ 9. Load the Generic FP16 bitstream

a. aocl program acl0 2-0-1_RC_FP16_Generic.aocx

Ensure programming was successful.

____ 10. Perform Diagnostics by running "aocl diagnose acl0"

Ensure you see the DIAGNOSTIC_PASSED message.

Step 2. Perform Inference with the FPGA

____ 1. Change directory into the <Lab Dir>/bin/intel64/Release

____ 2. Execute inference with the CPU

./demo -i dog2.jpg -l labels.txt -m breed_fp32.xml -d CPU

Notice the performance as well as the result.

____ 3. Perform the same inference with the FPGA using the 32bit network

./demo -i dog2.jpg -l labels.txt -m breed_fp32.xml -d HETERO:FPGA,CPU

Notice the confidence results, should expect slightly different result since the FPGA is doing the operations in FP16.

Also notice the performance.

Use the HETERO plug in to fall back to the CPU whenever a primitive is not supported on the FPGA. Here our network has a softmax layer that must be executed on the CPU.

____ 4. Perform the inference with the FPGA using the 16bit network

./demo -i dog2.jpg -l labels.txt -m breed_fp16.xml -d HETERO:FPGA,CPU

Notice the performance again. You should not see a difference in performance since the FPGA plugin simply truncates the values and the actual calculations are still done in FP16 on the FPGA as before.

____ 5. Lets try to run a more optimized 16bit FPGA image

a. pushd .

b. cd $IE_INSTALL/../a10_dcp_bitstreams

c. aocl program acl0 2-0-1_RC_FP16_GoogleNet.aocx

d. popd

Because our classification network is based on the GoogLeNet topology, we can use a more optimized FPGA image, removing the primitives that GoogLeNet doesn't need, allowing more Processing Elements to be placed onto the FPGA to accelerate convolutions.

____ 6. Perform the inference with the FPGA again

./demo -i dog2.jpg -l labels.txt -m breed_fp16.xml -d HETERO:FPGA,CPU

You should now see a significant increase in performance.

____ 7. FPGAs can also be configured with lesser data types for better performance, let's try to run the inference using FP11 Generic image

a. pushd .

b. cd $IE_INSTALL/../a10_dcp_bitstreams

c. aocl program acl0 2-0-1_RC_FP11_Generic.aocx

d. popd

____ 8. Run the inference again

./demo -i dog2.jpg -l labels.txt -m breed_fp16.xml -d HETERO:FPGA,CPU

You should see an performance increase again.

____ 9. Lets try the FP11 FPGA image optimized for GoogLeNet

a. pushd .

b. cd $IE_INSTALL/../a10_dcp_bitstreams

c. aocl program acl0 2-0-1_RC_FP11_GoogleNet.aocx

d. popd

____ 10. Run the inference again

./demo -i dog2.jpg -l labels.txt -m breed_fp16.xml -d HETERO:FPGA,CPU

As you can see due to the flexible nature of FPGAs, you'll see improvements in performance as you tailor the FPGA image to your network or by using lesser data types.

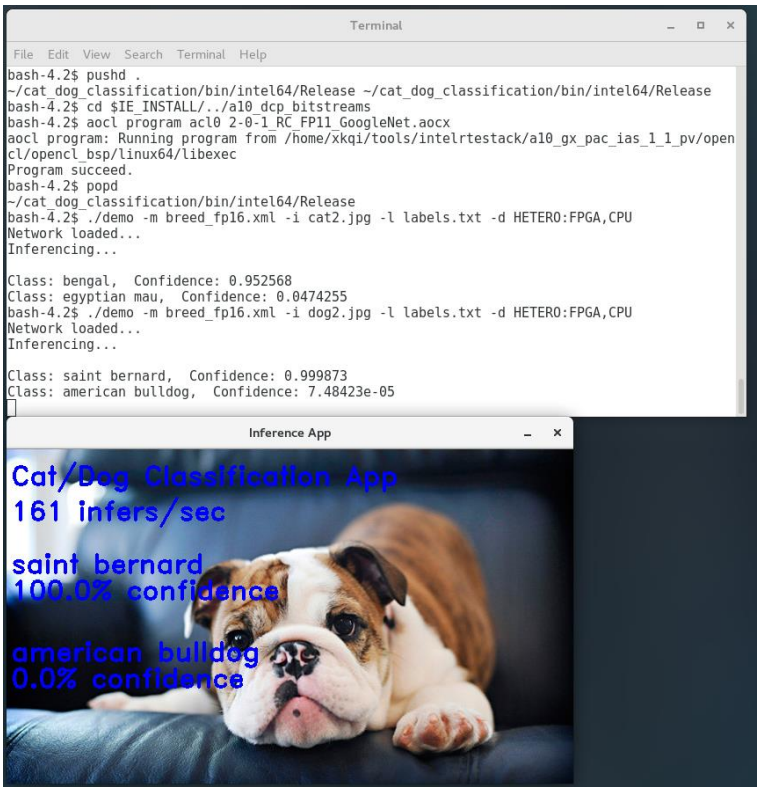


Figure 3. Dog classification results

Exercise Summary

- Practiced executing the same topology on various different FPGA images and compare performance results

END OF EXERCISE 2