Introduction to the Intel FPGA DevCloud

For Quartus Prime 18.0

1 Introduction

This tutorial provides an introduction to the cloud-based computing resource called the Intel® FPGA DevCloud.® Each computer in the DevCloud comprises both a high-end Intel CPU and an Intel FPGA device. The focus of this tutorial is on the development of Accelerator Functional Units (AFUs) for use on the DevCloud. An AFU is a hardware component that can be implemented in an FPGA and used to perform computations along with an Intel CPU. We show the required steps for setting up a secure connection to the DevCloud, and describe various commands that are available for AFU development. Detailed documentation on the Intel FPGA DevCloud can be found on *GitHub* at https://github.com/intel/FPGA-Devcloud.

Contents:

- Obtaining a user account on the Intel FPGA DevCloud
- Setting up a secure connection to the DevCloud using SSH
- Configuring the DevCloud for AFU development
- Compiling AFU hardware
- Programming an FPGA on the DevCloud
- Developing software programs that make use of an AFU

Requirements:

- Working knowledge of Microsoft Windows and/or Linux operating systems
- A computer running either Microsoft Windows (version 10 is recommended) or Linux (Ubuntu, or a similar Linux distribution). The computer would typically be either a desktop computer or laptop.
- Access to the Internet from your computer

2 Getting Started

Intel provides a set of tools called the *Intel Acceleration Stack*® for developing applications that use both Intel processors and hardware acceleration devices, such as FPGAs. The development tools provided by the acceleration stack are installed on the DevCloud, and include features for designing both hardware and software components. In this tutorial we show how to use these tools on the DevCloud for AFU development.

The first step to using the Intel FPGA DevCloud is to obtain a user account, which can be done by filling in the form at https://software.intel.com/en-us/devcloud/FPGA/sign-up. The application process takes about one to two weeks. Upon approval of your application an email message will be sent that provides a customized link for setting up your DevCloud access. The part of the email that you click on to reach the customized link is illustrated at the bottom of Figure 1. Clicking on this link takes you to a webpage that includes the connection link displayed in Figure 2. On this webpage you click on Connection Options to reach the selections displayed in Figure 3.

Dear User Name,

Welcome to Intel® DevCloud. Use this computing resource to develop, test, and run your workloads across a range of Intel® CPUs, GPUs, and FPGAs.

Your account has been activated and you can now connect to the Intel® DevCloud.

Go to the Intel DevCloud home page:

https://devcloud.intel.com/oneapi/?uuid=21da7c44-9c1f-48af-808f-a6b283d0534a

Figure 1. A part of the DevCloud welcome message.

Figure 2. Connect to the DevCloud.

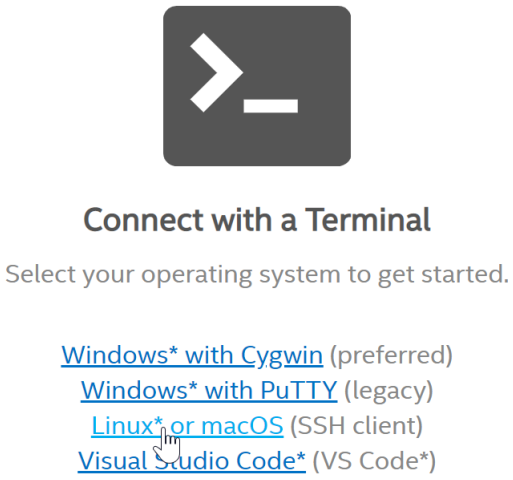

Figure 3. Connection options.

As indicated in Figure 3, you should click to select the item labeled Linux* or macOS (SSH client). This action opens a webpage customized to your DevCloud login information, as illustrated in Figure 4. The DevCloud account indicated in the figure, `u42132`, is shown for illustrative purposes only–the webpage is customized to set up each user's unique DevCloud account.

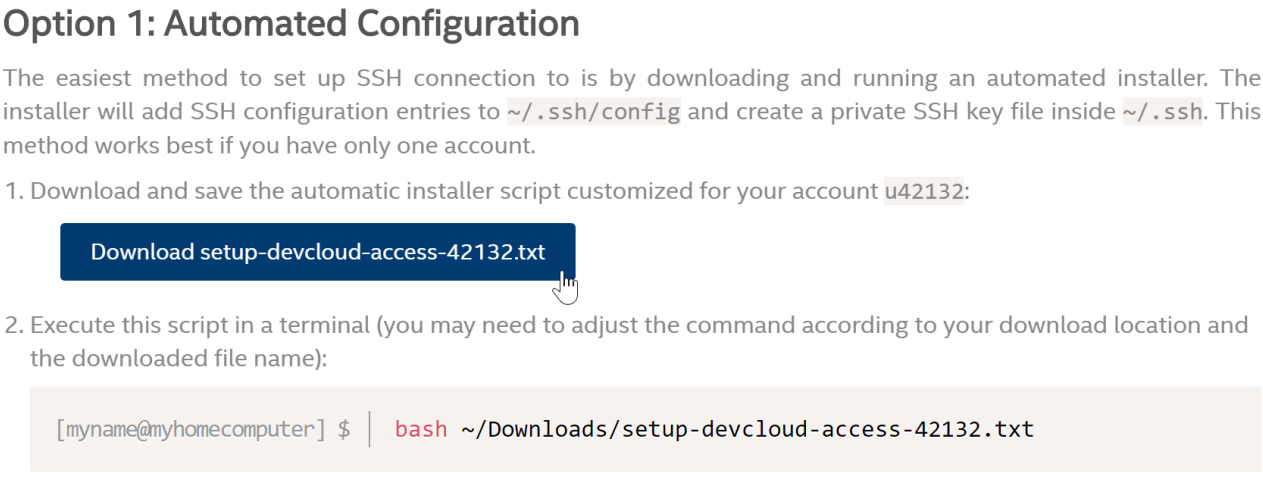

Figure 4. Setting up a secure connection to the DevCloud.

By clicking on the dark-blue box in "step 1" of Figure 4 you can download a file onto your home computer that contains your DevCloud secure login information, in *Secure Shell* (SSH) format. Before you can follow "step 2" of Figure 4 you have to be using a Linux environment, as discussed below.

2.1 Using Linux to Connect to the DevCloud

The SSH information downloaded using the link in Figure 4 requires a Linux environment. If you are already using a Linux system, then you can skip to Section 2.1.1. But if you are using Microsoft Windows, then a Linux environment has to be set up before continuing. We assume that you are running Linux via the Microsoft Windows extension called *Windows System for Linux* (WSL). If you are using another method of accessing Linux, for example the *Cygwin* tools, then some differences may apply in the setup process.

If not already done on your computer, enable WSL. Instruction for enabling this feature of Windows can be found by searching on the Internet. After WSL is enabled, download and install the `Ubuntu` app from the Microsoft Windows Store. Open the Ubuntu app, which provides you with a Linux terminal.

2.1.1 Installing SSH Authentication

To complete "step 2" of Figure 4, in a Linux terminal execute the command

```
bash PATH/setup-devcloud-access-``userid''.txt
```

where `PATH` represents the directory where the "setup" file was saved when it was downloaded in "step 1". This command extracts the SSH authentication information from the setup file and stores it into your Linux user's home directory in *~/.ssh*. As a final step modify a file permission by executing the command

```
chmod 600 ~/.ssh/config
```

Now, login to the DevCloud by executing the command

```
ssh devcloud
```

Once logged in, perform the following one-time set up. Edit the *.bashrc* file in your home directory on the DevCloud. Several *command-line interface* (CLI) text editors are available on the DevCloud, including *Vim*, *emacs*, *nano*, and *pico*. At the end of your *.bashrc* file append the line

```
source /data/intel_fpga/devcloudLoginToolSetup.sh
```

This setting ensures that your environment is properly configured to execute additional commands that are needed for AFU development. To apply this new setting to your login session, logout of the DevCloud by typing either the end-of-transmission character `^D` (hold down the `CTRL` key and press `d`), or `exit`, and then login again by executing `ssh devcloud`.

3 Using the DevCloud

The command `ssh devcloud` makes a connection to a DevCloud computer named *login-2*. This machine does not have access to any FPGA accelerator cards. To connect to a computer that has the required hardware resources execute (on *login-2*) the command

```
devcloud_login
```

This command allows you to make a connection to machines that offer five different types of compute-resources. For this tutorial we assume that AFU development will be done using an Arria 10 FPGA. If a different type of FPGA were to be used instead, then some of the instructions below would need to be modified accordingly. Type `1` to select

```
1) Arria 10 PAC Compilation and Programming - RTL AFU, OpenCL
```

You will then be presented with a choice of two versions of Arria 10 development tools. Type `1` to select

```
1) 1.2.1
```

You should now be connected to a computer that includes an Arria 10 FPGA card. In this discussion we assume that the computer is named *s005-n003*. To complete your setup, execute (on *s005-n003*) the command

```
tools_setup
```

This command presents seven options for selecting development software. Type `5` to choose

```
5) Arria 10 PAC Compilation and Programming - RTL AFU, OpenCL
```

At this point the environment should be configured for access to the AFU development tools. One of the software packages that we will use is called the *Open Programmable Accelerator Engine* (OPAE), which is part of the Intel Acceleration Stack. To verify the proper setup, type the command

```
echo $OPAE_PLATFORM_ROOT
```

This command should produce an output like the one illustrated in Figure 5. The OPAE software is installed on the DevCloud, and is also available in open-source form on *GitHub* at https://github.com/OPAE. This repository includes detailed documentation about OPAE.

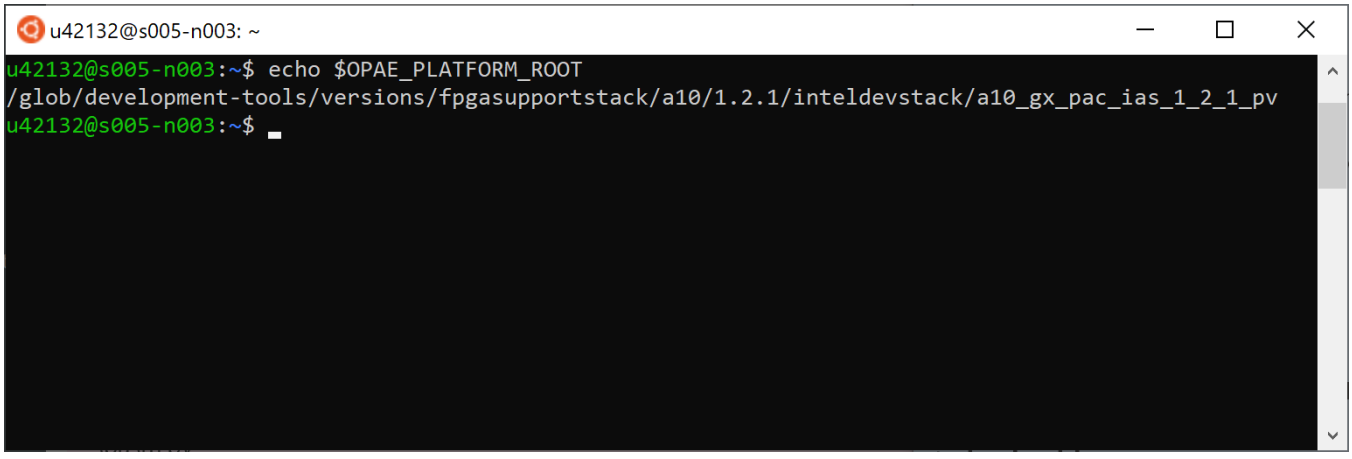

Figure 5. Verifying the proper setup.

3.1 Copying Files to/from the DevCloud

Files that are created on your home computer can be copied onto the DevCloud. One simple way to perform the desired file-transfer is to use the *Secure Copy* (`scp`) program. For example, to copy *homefile* from your computer to your *home* directory (~) on the DevCloud you would use the command

```
scp homefile devcloud:~/
```

Of course, you can copy files to other directories on the DevCloud by including the desired path in the `scp` command. Similarly, `scp` can be used to copy a file from the DevCloud to the your home computer. For example, to copy *devfile* from the DevCloud to the current working directory (`.`) on your home computer you would use the command

```
scp devcloud:~/devfile .
```

The `scp` command can by used to copy an entire directory *tree*, including sub-directories, by using the `-r` option. More information about `scp` can be found by typing `man scp`. Note that `scp` should be used with some caution, as it *clobbers* (overwrites) existing files without warning. Thus, if you copy a file to a remote computer using `scp`, the file is overwritten on the remote computer if it already exists.

3.2 Working with an AFU

Intel documentation specifies that the source files for an AFU have to be structured in a particular way. An example of a simple AFU, named *hello_afu*, is provided as a sample on the DevCloud in the directory

```
$OPAE_PLATFORM_ROOT/hw/samples
```

The AFU has a root directory called `hello_afu`, which contains directories named `hw` and `sw`. The `hw` directory contains a directory called `rtl`, which holds the hardware source-code files. Figure 6 (near the top) shows the contents of the sample `rtl` folder. If you create a new AFU, then before it can be compiled for the first time you have to set your working directory to the root, such as `hello_afu`, and then execute the command

```
afu_synth_setup --source hw/rtl/filelist.txt build_synth
```

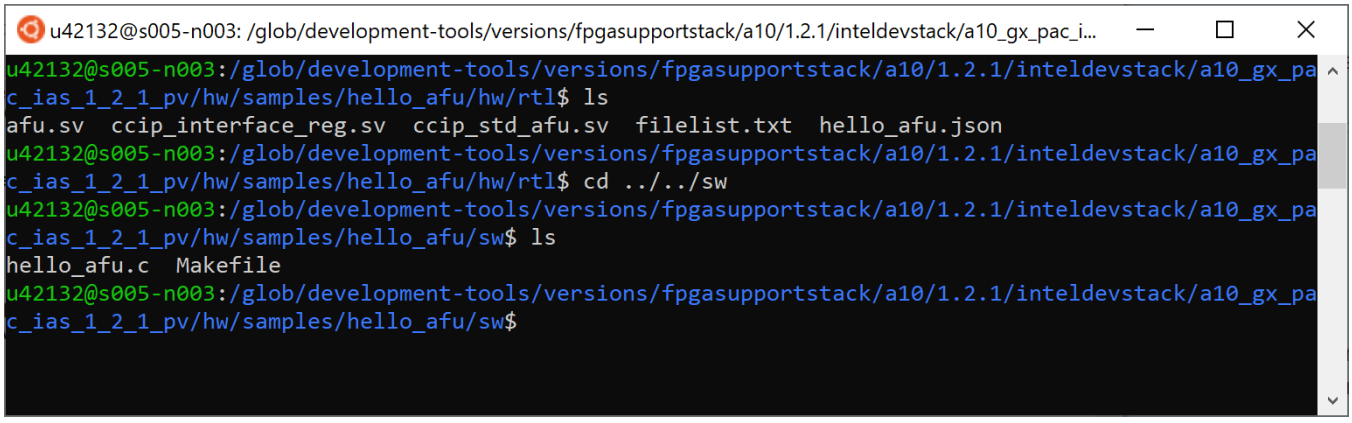

Figure 6. Sample AFU hardware source-code files.

This command creates the directory named `build_synth` inside the `hello_afu` directory, and copies a number of files that are needed to compile the AFU into a hardware circuit. To perform the hardware compilation you would set your working directory to `build_synth` and execute the command

```
run.sh
```

This command, which is found in the directory `$OPAE_PLATFORM_ROOT/bin`, runs the tools required to generate a hardware circuit for the AFU. In the case of the *hello_afu* example the circuit would be saved in a bitstream file called *hello_afu.gbs*.

Some of the files created during hardware compilation can be "cleaned up" by executing in the `build_synth` directory the command

```
clean.sh
```

3.2.1 Downloading an AFU Bitstream into an FPGA

You can download a bitstream file such as *hello_afu.gbs* into an FPGA using a two-step process. First, execute the command

```
PACSign PR -t UPDATE -H openssl_manager -i hello_afu.gbs -o hello_afu_unsigned.gbs
```

Then, execute

```
fpgasupdate hello_afu_unsigned.gbs
```

3.2.2 Compiling Software Programs for an AFU

Software programs that utilize an AFU have to be contained in its `sw` directory. These files can be compiled into executable programs by using a special *Makefile*. An example Makefile, as well as a C source-code file, can be found in the `sw` folder for *hello_afu*, as illustrated in Figure 6 (near the bottom).

3.2.3 Miscellaneous Commands

Some miscellaneous Linux and DevCloud <*commands*> are described below.

< ! > You can use `!` to recall a previously executed command. For example, to re-execute a previous `ssh` command you can type `!shh`. The Linux shell will search your command history for the last command that started with *ssh* and execute it again.

< !:p > This command is similar to `!`, except that is does not *execute* the recalled command. For example, if you wish to check for the most-recently executed command that begins with *ssh*, you would type `!ssh:p`.

< ↑ > You can use ↑ to recall previously-executed commands.

< ^Z > When you are using a program, for example a text editor, typing ^Z (hold down the `CTRL` key and press `z`) returns control to the Linux command-line prompt, but leaves your program running in the *background*.

< ~^Z > When you are connected to a *remote* computer, typing the character sequence ~^Z returns control to your *local* computer, but keeps the session on the remote computer running in the *background*.

< bg > This command displays a list of all programs that you are running in the background.

< fg > The `fg` command returns control to the most recently-suspended background program. If there are multiple background programs then you can return control to program *n* by executing `fg %n`.

< ps > This command provides a list of currently-executing processes.

< kill > The `kill` command can be used to terminate a process.

< killall > Occasionally you may get automatically logged out of a compute-machine on the DevCloud, for example the machine *s005-n003*, even though you have not purposely logged out. If this occurs then a subsequent issue may develop. Specifically, trying to reconnect to a compute-machine using the command `devcloud_login` may result in the error

```
You are already logged into node =s005-n003 interactively.
```

In this case, you can reconnect to the machine by executing

```
ssh s005-n003
```

After reconnecting, in some situations you may find that commands on the DevCloud do not work properly. If this happens, then (as a last resort) you can kill all processes owned by your *userid*, by using the `killall` command. For example, if your userid were *u42132* you would type

```
killall --user u42132
```

This command should terminate all processes owned by you and then disconnect from the machine. If you do not get logged out, then execute the command one more time.

< uuidgen > This DevCloud command generates a *universally unique identifier* (uuid) for an AFU.

< pbsnodes > This DevCloud command provides a listing of available compute-machines. To control the display of this list use the command `pbsnodes | more`.

4 Concluding Remarks

This tutorial has provided an introduction to the Intel FPGA DevCloud for AFU development. Further details about the DevCloud can be found on its *GitHub* site at https://github.com/intel/FPGA-Devcloud.