EENG 428 / ENAS 968

Cloud FPGA

Prof. Jakub Szefer

Getting Started With Amazon EC2 F1 Compute Instances: Hello World Example

Authors:
Jakub Szefer and Victor Liu
Yale University
New Haven, CT 06511, USA

Revised:
October 19, 2019

Contents

1 Introduction

This document covers basic steps needed to get started with Amazon EC2 F1 compute instances. It covers information about setting up Amazon EC2 account, creating and connecting to development VMs, creating F1 instances, and running basic *hello world* demo provided by Amazon, which involves both hardware FPGA Custom Logic (CL) design loaded onto F1 instance's FPGA, and corresponding software code running on F1 instance that communicates with the hardware design.

1.1 Prerequisites

Working with Amazon EC2 F1 compute instances requires:

- **Computer with SSH client**: The virtual machines (VMs) running in Amazon's cloud are typically accessed via a remote SSH connection. Standard ssh utility on Linux or Mac, or PuTTY on Windows are needed to establish the connections and access the VMs.
- **Amazon EC2 account**: An individual account on Amazon EC2 is required. The account is typically created and accessed through Amazon's web page at `https://www.amazon.com/ec2`.
- **Access to aws-fpga git repository**: Amazon maintains a repository on GitHub where relevant code and scripts are located, source code for the examples is also provided through the GitHub repository. It can be accessed at `https://github.com/aws/aws-fpga`. This repository is open to anybody.

1.2 Amazon EC2 Account and Consolidated Billing

Amazon charges per hour (or per minute) for running VMs in the cloud, the Amazon EC2 F1 instances are no different. Each user needs to provide credit card number when registering with Amazon, all charges are billed monthly to the credit card.

For projects related to EENG 428 work, *consolidated billing* feature is used, where expenses incurred by the user are charged to EENG 428 account, not to user's credit card. Upon creating Amazon EC2 account, account number should be sent to the instructor so that the account can be set up with the consolidated billing feature. Account ID can be found on `https://console.aws.amazon.com/billing/home#/account`.

2 Creating Development VM Instance

This section covers steps needed to create a `c4.4xlarge` instance that will be used to synthesize the hardware design (and optionally in future compile C code for the software).

2.1 Create `c4.4xlarge` Instance

The development, compilation and synthesis of the C code and FPGA code is done using a virtual machine (VM) running on a `c4.4xlarge` instance in Amazon's cloud computing data centers. To create a development VM for working with C and FPGA code, sign into the AWS console: `https://console.aws.amazon.com`. In the top-right corner make sure you're in `US East (N. Virginia)` region, Amazon EC2 F1 compute instances are not available in all regions[1].

From AWS console go to the `EC2 Dashboard` and then go to `Instances` page. Now you can select `Launch Instance` and begin the process of configuring and launching an Amazon EC2 F1 compute instance. Note, after each step press `Next` until you get to Step 7, do not skip steps and do not press `Review and Launch` before Step 7.

- Step 1 – Chose AMI: Search for "FPGA" in the AWS Marketplace and select the "FPGA Developer AMI." This AMI contains all the licensed software needed to synthesize the FPGA code.
- Step 2 – Choose Instance Type: The recommended instance type is `c4.4xlarge` for synthesizing your FPGA logic, also called Custom Logic (CL) by Amazon. Other instance types can be used, but CPU, memory, and disk type will affect synthesis speed.
- Step 3 – Configure Instance: Default configuration should be sufficient, but ensure you are only launching 1 instance.
- Step 4 – Add Storage: Change the size of the root volume to 128 GiB, to ensure sufficient space to download and synthesize code.
- Step 5 – Add Tags: No actions needed in this step.
- Step 6 – Configure Security Group: No actions needed in this step. However, ensure only port 22 (SSH) is open.
- Step 7 – Review Instance Launch: Press `Launch` button to begin launching the instance.

2.2 Selecting Key Pair and Launch the Instance

Instead of passwords, the virtual machines (VMs) in Amazon's cloud deployments use cryptographic keys for authenticating users and letting you log into the virtual machine via SSH connection. A cryptographic key pair consists of a public key that AWS stores, and a private key that you store (in a file). Together, they allow you to connect to your instance securely.

If you don't have an existing key file, or if the file is lost, select `Create a new key pair` in the `Select new key pair or create a new key pair` window. Then, enter a name for the key and select `Download Key Pair` to download the key (as a `.pem` file) to your computer. After downloading the file, ensure to change permissions so only you, and not other users on your computer, can access the file. Anybody with access to the file, or copy of the file, can log into your virtual machines. The file needs to be protected. On Linux, you can use command: `chmod 400 *.pem`.

[1]Also, selecting instance in region which is physically close to your location should reduce network delays and improve connection speed.

Finally, press `Launch Instances` to start the virtual machine. You can monitor the status of the instance on the `EC2 Dashboard` on the `Instances` page.

2.3 SSH'ing into the VM

Once the VM has started and its status has changed to `running` on the `EC2 Dashboard` on the `Instances` page, you can select the instance, then press `Actions` and then press `Connect`. Each time you start the VM, it will get a different IP address, so you need to check the `Connect` window to see the proper IP and command line to connect to the instance. The `Connect` window gives information how to connect to the instance using SSH and the `*.pem` file that contains your secret key.

Note, for Linux users, you are given command line you can cut-and-paste straight into terminal. Windows (PuTTY) users need to follow extra information that is given in the `Connect` window.

Further, note that the user name you are using to connect to the VM is by default `root`. However, for the "FPGA Developer AMI" the proper user name is `centos` (the VM uses CentOS Linux operating system distribution). There should be no password associated with the `centos` user, and the user has root privilege. E.g. you can execute `sudo` commands without password, such as `sudo yum install vim` to install software such as the Vim text editor.

2.4 Setting up AWS CLI and S3 Bucket to Enable AFI Creation

As the developer you are required to create a S3 bucket for the AFI (Amazon FPGA Image) generation. A S3 bucket is similar to a file stored on a remote server, the files are not stored inside the VM instance, rather they are stored remotely in Amazon's cloud. The bucket will contain a `*.tar` file and logs which are generated from the AFI creation service. To be able to create S3 buckets from command line, and to do other AWS related operations from the command line, the AWS CLI has to be first configured.

2.4.1 Setup AWS CLI Access

AWS CLI (Command Line Interface) tools need to know your credentials so that when you execute AWS CLI commands. You can configure the AWS CLI using below command:

```
$ aws configure
```

You will need your security credentials. Go to `https://console.aws.amazon.com/iam/home?#/security_credential`. You may need to select `Continue to Security Credentials` in a pop-up window. Expand the `Access keys (access key ID and secret access key)` tab. Click `Create New Access Key`. Make sure to download the access key file, you will not have access to the secret key after you finish the access key creation process. The file will be a `*.csv` (comma separated values file type) which can be opened with most spreadsheet programs or text editors. Save the file for future access.

In the `aws configure` command, you will need to enter your access key and the corresponding secret key. Also, make sure to set the region to `us-east-1` and select output format as `json`.

Note that the input text is are case-sensitive, e.g. `us-east-1` and not `US-East-1` You can re-run the command if you make any mistakes.

2.4.2 Create S3 Bucket

This S3 bucket will be used by the AWS SDAccel scripts to upload your DCP (digital checkpoint) to AWS for AFI generation. Start by creating a bucket and a folder within your new bucket, details about the commands are below.

The first command creates an S3 bucket, which needs to have a unique bucket name among all of buckets in S3. The file name cannot have upper-case letters or underscores. Replace the <bucket-name> with the actual bucket name you want to use. Do not use < or > in the actual name. The name can only use lower-case letters and dashes. The name needs to be globally unique, so pre-pending the name with "eeng428" can help you have a useful, but unique name for your bucket.

```
$ aws s3 mb s3://<bucket-name> --region us-east-1
$ aws s3 mb s3://<bucket-name>/<dcp-folder-name>
$ touch FILES_GO_HERE.txt
$ aws s3 cp FILES_GO_HERE.txt s3://<bucket-name>/<dcp-folder-name>/
```

The second command creates a folder within the bucket. Replace the <dcp-folder-name> with the actual folder name. Again, do not use < or > in the actual name of the folder.

The third command simply creates a blank file in the local directory, not in bucket, yet. The fourth command copies the blank file into the DCP folder in the bucket, which forces the bucket's folders to be actually created. Don't forget the trailing slash at end of last command.

The AFI creation process will also generate logs that will be placed into your S3 bucket. These logs can be used for debug if the AFI generation fails. Thus, create a folder for your log files:

```
$ aws s3 mb s3://<bucket-name>/<logs-folder-name>
$ touch LOGS_FILES_GO_HERE.txt
$ aws s3 cp LOGS_FILES_GO_HERE.txt s3://<bucket-name>/<logs-folder-name>/
```

The first command creates a folder to keep your logs. The second command creates a blank file again. The third command copies blank file into the bucket and trigger creation of the folder on S3.

The same bucket can be re-used among projects or as you re-synthesize the design, e.g. create different DCP sub-folders for each project. Alternatively, you can create separate buckets for each project to keep things organized.

2.5 Installing the HDK and Setting up the Environment

The hardware development kit (HDK) contains all the scripts and links to tools needed to synthesize your Custom Logic designed in VHDL or Verilog into the digital checkpoint (DCP) that will be used to finally create the Amazon FPGA Image (AFI) that can be used to configure the FPGA.

2.5.1 Get and Setup Amazon AWS FPGA Code

First, `git clone` a copy of aws-fpga repository in the home directory for the VM.

```
$ git clone https://github.com/aws/aws-fpga
```

Next, enter the aws-fpga directory and source hdk_set.sh.

```
$ cd aws-fpga/
$ source hdk_setup.sh
```

Sourcing hdk_setup.sh will set required environment variables that are used throughout the examples in the HDK. Optionally, source software development kit (SDK) setup script to get access to scripts about AFI's created or scripts to get e-mails for AFI generation updates.

```
$ source sdk_setup.sh
```

You may need to source the HDK and SDK scripts each time you log into the VM.

2.6 Select Hello World Example for Synthesis

Select the Hello World example to be your Custom Logic directory and set your email address to receive notification. Note, sometimes the notification by e-mail does not work, usually if there is some error in compilation or environment is not setup correctly. Always check manually if the compilation is done.

```
$ export HDK_DIR=$(pwd)/hdk
$ cd $HDK_DIR/cl/examples/cl_hello_world
$ export CL_DIR=$(pwd)
$ export EMAIL=your.email@example.com
```

Setting up the CL_DIR environment variable is crucial as the build scripts rely on that value to locate the code that should be synthesized. Each CL_DIR follows the recommended directory structure to match the expected structure for HDK simulation and build scripts. The EMAIL environment variable is used to notify you when your Custom Logic build has completed.

Note, first time you synthesize a design and request an e-mail, you may get an e-mail from Amazon about *AWS Notification - Subscription Confirmation*. You need to confirm your subscription to the notification e-mails before you will actually be able to receive them.

2.7 Start the Build Process for the Custom Logic

Executing the `aws_build_dcp_from_cl.sh` script will perform the entire synthesis process converting the CL design into a completed Design Checkpoint that meets timing and placement constrains of the target FPGA. The output is a tarball file comprising the DCP file, and other log/manifest files, formatted as `YY_MM_DD-hhmm.Developer_CL.tar`. This file would be submitted to AWS to create an AFI. By default the build script will use Clock Group A Recipe A0 which uses a main clock of 125 MHz. I.e. your custom logic will be running at 125 MHz.

Start the build process:

```
$ cd $CL_DIR/build/scripts
$ ./aws_build_dcp_from_cl.sh -notify
```

Once the build process is started, you may disconnect from the VM (e.g. end the SSH session), and re-connect later to check on the build process or to continue with the generation of the AFI image if the build finished successfully. Note, however, the VM cannot be stopped so that the compilation process continues. Also note, the design may take few hours to synthesize – once done, the VM can be stopped if you don't want to immediately proceed with the AFI creating steps.

2.8 Monitoring Build Process

Since the DCP generation can take up to several hours to complete, hence the `aws_build_dcp_from_cl.sh` will run the main build process (vivado) in within a `nohup` context: This will allow the build to continue running even if the SSH session is terminated half way through the run, for example. The `-notify` flag indicates to the script to notify you at your EMAIL address when the build is completed.

If you do not receive the notification e-mail, check if the `*.tar` file is generated in `$CL_DIR/build/checkpoints/` directory.

If the compilation fails, no `*.tar` file will be generated. You can find logs of the most recent compilation in the `$CL_DIR/build/scripts` folder. You may need to read the log files to find the source of the error.

When build process is done successfully, the final `*.tar` file should be available in sub directory of CL_DIR:

```
$ cd $CL_DIR/build/checkpoints/to_aws
```

2.9 Submit the DCP Tarball to AWS to Create the AFI

Once you have the `*.Developer_CL.tar`, which contains the DCP (Design Checkpoint), you can being AFI creation. To submit the DCP, use the S3 bucket and upload the tarball file into that bucket. You need to prepare the following information:

1. Name of the logic design (Optional).
2. Generic description of the logic design (Optional).
3. Location of the tarball file within S3 bucket.
4. Location of an S3 directory where AWS would write back logs of the AFI creation.
5. AWS region where the AFI will be created. An AFI needs to be created in same region as the F1 instance that will use it. Use `copy-fpga-image` AWS CLI command to copy an AFI to a different region if needed.

To upload your tarball file to S3, use the bucket and folder you created initially for your tarball, then copy files into S3 using a single command:

```
$ aws s3 cp $CL_DIR/build/checkpoints/to_aws/*.Developer_CL.tar \
  s3://<bucket-name>/<dcp-folder-name>/
```

Note, the trailing '/' is required after <`dcp-folder-name`>. Also, if you have logged out (while the design synthesizes) you may need to `export` the $CL_DIR variable again, as the environment variables are reset each time you log back in.

2.10 Start AFI Creation

Use the `create-fpga-image` command to send the tarball file to Amazon for AFI creation.

```
$ aws ec2 create-fpga-image \
    --region <region> \
    --name <afi-name> \
    --description <afi-description> \
    --input-storage-location Bucket=<dcp-bucket-name>,Key=<path-to-tarball> \
    --logs-storage-location Bucket=<logs-bucket-name>,Key=<path-to-logs> \
    [ --client-token <value> ] \
    [ --dry-run | --no-dry-run ]
```

Note, <`path-to-tarball`> is your <`dcp-folder-name`>/<`tar-file-name`>, while <`path-to-logs`> is your <`logs-folder-name`>.

The output of this command includes two identifiers that refer to your AFI:

- FPGA Image Identifier or AFI ID: this is the main ID used to manage your AFI through the AWS CLI commands and AWS SDK APIs. This ID is regional, i.e., if an AFI is copied across multiple regions, it will have a different unique AFI ID in each region. An example AFI ID is afi-06d0ffc989feeea2a.
- Global FPGA Image Identifier or AGFI ID: this is a global ID that is used to refer to an AFI from within an F1 instance. For example, to load or clear an AFI from an FPGA slot, you use the AGFI ID. Since the AGFI IDs is global (by design), it allows you to copy a combination of AFI/AMI to multiple regions, and they will work without requiring any extra setup. An example AGFI ID is agfi-0f0e045f919413242.

The `create-fpga-image` command submits your Custom Logic to Amazon for approval. This usually takes about 20-30 minutes.

The `describe-fpga-images` API allows you to check the AFI state during the background AFI generation process. You must provide the FPGA Image Identifier returned by `create-fpga-image`:

```
$ aws ec2 describe-fpga-images --fpga-image-ids afi-06d0ffc989feeea2a
```

You can use the `wait_for_afi.py` script to wait for the AFI creation to complete and get an email with the results. The AFI can only be loaded to an instance once the AFI generation completes and the AFI state is set to `available`.

3 Creating F1 VM Instance with Access to FPGA

Once an AGFI ID is available, the design can be loaded and tested on an FPGA. For this purpose you need to start a new VM, and Amazon EC2 F1 instance, that has access to FPGAs.

3.1 Requesting Access to F1 Instances

By default, Amazon EC2 users may not have access to F1 instances. To get access to F1 instances, you need to request a "limit increase" for F1 instances.

First, log into the AWS EC2 Dashboard, at `https://console.aws.amazon.com/ec2`. On the left-hand side select `Limits` under the `EC2 Dashboard` heading. Next, search for `f1.2xlarge` row in the table and click `Request limit increase`.

On the request form, fill in:

- Limit Type = EC2 Instances
- Region = US East (Northern Virginia)
- Primary Instance Type = f1.2xlarge
- Limit = Instance Limit
- New limit value = 1 (or 2 or more if you want to run more FPGAs in parallel in future[2])
- Use Case Description = Mention this request is for class or research project.
- Support Language = English (or pick your language)
- Contact method = Web or Phone as you wish

Finally, click `Submit` to submit the request, request should be processed in few hours.

3.2 Launch an F1 Instance

An Amazon EC2 F1 instance is needed to run the AGFI on an FPGA.

Follow the same procedure as launching the `c4.4xlarge` instance with the FPGA Developer AMI, except this time with a `f1.2xlarge` instance.

Note, curiously `f1.2xlarge` gives access to 1 FPGA, while `f1.16xlarge` gives access to 8 FPGAs. While the naming is confusing, use `f1.2xlarge` as only 1 FPGA board is needed.

3.3 SSH'ing into the VM

Follow previous instructions to SSH into the F1 VM.

Note, you may thus now have two VMs running in parallel and have two SSH connections, one for each VM.

[2]You may want to keep the number small as a safety feature so you do not accidentally start too many FPGA instances in parallel and incur a large cost.

3.4 Setup the FPGA Management Tools and Configure AWS CLI

Similar to setup of the development VM, you need to download the `aws-fpga` code form Amazon's git repository. Further, you need to setup the SDK and configure the CLI.

Download `aws-fpga` code again.

```
$ git clone https://github.com/aws/aws-fpga
```

Then, source the SDK to setup environment. Sourcing sdk_setup.sh installs and sets up all the FPGA Management tools necessary for loading and testing your AFI.

```
$ cd aws-fpga/
$ source sdk_setup.sh
```

Only SDK is needed on F1 since you're not synthesizing the code, only loading already generated code onto the FPGA.

Furthermore, you need to configure AWS CLI again, since this time you're working in a new VM, the F1 instance. As before use `aws configure` to provide necessary information. Again, use `us-east-1` region and `json` as output file type.

```
$ aws configure
```

3.5 Load the AGFI onto FPGA

You can now use the FPGA Management tools, from within your F1 instance, to load your AFI onto an FPGA on a specific slot. Since we are using F1 instance with 1 FPGA, there is only 1 slot, other instance types will have more slots.

Make sure you clear any AFI you have previously loaded in your slot:

```
$ sudo fpga-clear-local-image -S 0
```

Now, load your AFI to FPGA slot 0. Make sure to change the AGFI ID to the actual AGFI ID for your generated design.

```
$ sudo fpga-load-local-image -S 0 -I agfi-0fcf87119b8e97bf3
```

You can verify that the AFI was loaded properly if the output shows the FPGA in the `loaded` state after the FPGA image load operation.

3.5.1 Pre-synthesized Hello World Example

Amazon maintains pre-synthesized hello world example. Pre-generated Hello World example can be accessed using public AGFI number. Pre-generated AFI ID for hello world example is `afi-03d11a4ea66e883ef`. Pre-generated AGFI ID for hello world example is `agfi-0fcf87119b8e97bf3`.

You can use these to compare behavior of your generated example with the pre-generated examples.

3.6 Validating the Design Using the CL Example Software

Each CL example comes with a runtime software under `$CL_DIR/software/runtime/` subdirectory. The `$CL_DIR` is the directory of the custom logic you are using, for the hello world example it will typically be in the examples directory.

```
$ cd /home/$USER/aws-fpga/hdk/cl/examples/cl_hello_world
$ export CL_DIR=$PWD
```

Once you locate the hello world example, you will need to build the runtime application that matches your loaded hello world AFI.

```
$ cd $CL_DIR/software/runtime/
$ make all
$ sudo ./test_hello_world
```

To execute `./test_hello_world` application you need sudo privilege as the code requires access to the PCI express bus, which is usually restricted. Within the example software you can use different commands to communicate with the example hardware hello world CL loaded on the FPGA and to get some outputs form the CL on the FPGA.

3.7 Allow Other Users to Access Your AGFI

Once an AGFI is created, it can be used by other users, or by instructor to test your code. To allow others to access your AGFI, you need to change the permissions. There is a modify load permissions command which can be run as follows to allow a specific user access to your AGFI

To add user ID (AWS account ID) with permissions to load a specific AGFI, use following command. Make sure to change the AFI ID, and the user ID as needed.

```
$ aws ec2 --region us-east-1 modify-fpga-image-attribute \
  --fpga-image-id afi-0e5361a69d2af203d \
  --operation-type add --user-ids 095707098027
```

The output of the command should be similar to:

```
{ "FpgaImageAttribute": { "FpgaImageId": "afi-0e5361a69d2af203d",
"LoadPermissions": [ { "UserId": "095707098027" } ] } }
```

Note that above will add the permissions to the AFI not the AGFI. But the permissions for the AFI will allow access to the AGFI as well. There is also a command to make the AFI and AGFI public which can be found at `https://github.com/aws/aws-fpga/blob/master/hdk/docs/fpga_image_attributes.md`.

Upon completion of this tutorial, add the load permissions for the instructor so your compiled *hello world* example can be tested. Instructor's AWS account ID will be provided to the class by e-mail.

3.8 Shutting Down VM when Done Working

When you are done working, or are taking a break, make sure to shut down the VMs, as the VM usage is charged by hour, independent of whether you are actually using the VM or the VM has just been left running idle. Not, do not stop the VM while synthesizing the FPGA code, as that will stop the build process, on the other hand, once the tarball is uploaded into S3 bucket, the VM can be shutdown while waiting for Amazon to approve the design.

Before shutting down the VM, make sure to save all the files, upload any modified files to git, and then log out. Also, clear the FPGA slot before shutting things down if this is an F1 instance.

To shut down the VM, go to the `EC2 Dashboard` and the `Instances` page. Select the instance, then press `Actions` and then press `Instance State` and then press `Stop`. Once the VM has stopped its status will change to `stopped`. Shutting down the VM is similar to turning off the computer, next time you start up the VM, all the files and configuration should be there (except data or files stored to the ephemeral drive, which is erased each time).

4 Acknowledgement

This tutorial document was based on information available from Amazon in their documentation. The authors would like to thank Shanquan Tian, Adam Wolnikowski, and Samantha Bleykhman for testing the tutorial and providing feedback.