**Section I: Amazon AWS Instance Setup**

1. Create an account and log in the amazon AWS service, at <https://aws.amazon.com>. A form of payment would be required if an FPGA is not labeled to be free.
2. Select EC2 service once logged in.
3. Select “Launch Instance” blue tab under the “Create Instance” potion of the page.
4. Select the Amazon Linux AMI for the instance
5. Select the instance type: f1.2xlarge, a cost will be applied for the use of this instance type
6. Review your Instance configuration and then Launch the instance

**Section II: Private Key Creation**

1. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>
2. In the navigation pane, under “Network & Security”, choose “Key Pairs”. The navigation pane is on the left side of the Amazon EC2 console. If you do not see the pane, it might be minimized; choose the arrow to expand the pane.
3. Choose “Create Key Pair”.
4. Enter a name for the new key pair in the “Key pair name” field of the “Create Key Pair” dialog box, and then choose “Create”.
5. The private key file is automatically downloaded by your browser. The base file name is the name you specified as the name of your key pair, and the file name extension is .pem. Save the private key file in a familiar location in your file system. IMPORTANT: This is the only chance for you to save the private key file. You'll need to provide the name of your key pair when you launch an instance and the corresponding private key each time you connect to the instance.
6. This Private Key will be used as a SSH client to connect to your Linux instance, use the following command to set the permissions of your private key file so that only you can read it.

**Section III: Using Putty to SSH to the instance**

1. The Instance status should be visible in the Instance Tab of the EC2 dashboard. Select “Instances” and it would show all the current instances that are running in your account.
2. In the “Public DNS” of the running instance, copy the information presented, it should begin with ec2. Copy this.
3. Open the Putty application.
4. In the Host Name box type in “ec2-user@”. Following the @ paste the Public DNS that was copied in step 2.
5. Now in the Category portion of Putty, open the SSH tab, and select the Auth tab. Under the “Authentication parameters” of the Auth tab, a browse button should be visible. Search from the Private Key that was created in Section II.
6. Select the “Open” on the bottom right of the screen.

**Section IV: Running an Amazon FPGA Image (AFI)**

In this section, you will learn how to load and run a pre-generated AFI for the Hello World example code which is located at:

<https://github.com/aws/aws-fpga/tree/master/hdk/cl/examples/cl_hello_world_vhdl>

1. After logging into the F1 instance using Putty, you will need to install a git revision control utility.

$ sudo yum install git

1. Download the AWS FPGA repository using the git function. CD into the aws-fpga directory created.

$ git clone https://github.com/aws/aws-fpga.git $AWS\_FPGA\_REPO\_DIR

$ cd aws-fpga

1. Download the Development Tools which will be needed to run the sdk setup shell script.

$ sudo yum groupinstall ‘Development Tools’

1. Run the sdk\_setup.sh script. This will setup the commands needed to clear and load an AFI onto the FPGA.

$ . sdk\_setup.sh

1. To clear the FPGA image, use the FPGA clear local image tool. The -S 0 selects which FPGA to clear in the case that you are running the F1 instance that has multiple FPGAs.

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

1. To see the status of the FPGA in the system, use the FPGA describe local image tool.

$ sudo fpga-describe-local-image -S 0 -H

1. Now we can load the image using the FPGA load local image tool. The -I selects the AFI. In this case a pre-generated AFI for the Hello World example used in this tutorial is provided.

$ sudo fpga-load-local-image -S 0 -I agfi-0F0E045F919413242

1. Now that the AFI is loaded, you can run the software code located in the software/runtime folder. CD into this folder as shown below.

$ cd hdk/cl/examples/cl\_hello\_world\_vhdl/software/runtime/

1. To build use the make all function. Then run the C code.

$ make all

$ sudo ./test\_hello\_world

**Section V: Building an AFI**

In this section, you will learn how build an AFI for the Hello World example code which is located at:

<https://github.com/aws/aws-fpga/tree/master/hdk/cl/examples/cl_hello_world_vhdl>

1. You will need to login to an instance launched by the FPGA Developer AMI from the link below. Select your region then select the instance you want to use. t2.micro is the free instance. However, t2.2xlarge is recommended to reduce the amount of time to build your AFI.  
     
   <https://aws.amazon.com/marketplace/fulfillment?productId=40257ab5-6688-4c95-97d1-e251a40fd1fc&ref_=dtl_psb_continue&region=us-east-1>
2. When using Putty to SSH into this instance, use the username centos instead of ec2-user. Once successfully logged into the terminal, download the AWS FPGA repository then CD into the aws-fpga directory created.  
     
   $ git clone https://github.com/aws/aws-fpga.git  
   $ cd aws-fpga
3. Source the hdk setup shell script.  
     
   $ source hdk\_setup.sh
4. Set your AWS credentials. Supported regions are us-east-1, us-west-2, and eu-west-1.  
     
   $ aws configure
5. Navigate to the Hello World directory and set it to be the CL\_DIR directory.  
     
   $ cd $HDK\_DIR/cl/examples/cl\_hello\_world\_vhdl  
   $ export CL\_DIR=$(pwd)
6. Navigate to the build/scripts folder and run the aws\_build\_dcp\_from\_cl shell script. This will build the AFI and can take a couple hours to complete. You can set up your e-mail address to get a notification of when the build is done by using the export command and using the -notify argument when running the script. You can also have the script run in the foreground by using the -foreground argument.   
     
   $ export EMAIL=your.email@example.com  
   $ cd $CL\_DIR/build/scripts  
   $ ./aws\_build\_dcp\_from\_cl.sh -notify -foreground
7. Once the build is complete, you will need to create a bucket and folder for the generated tarball, then upload it to S3.  
     
   $ aws s3 mb s3://<bucket-name> --region <region>  
   $ aws s3 mb s3://<bucket-name>/<dcp-folder-name>/  
   $ aws s3 mb cp $CL\_DIR/build/checkpoints/to\_aws/\*.Develoer\_CL.tar \  
    s3://<bucket-name>/<dcp-folder-name>/
8. Create the ID for your AFI.  
     
   $ aws ec2 create-fpga-image \  
    --region <region> \  
    --name <afi-name> \   
    --description <afi-description> \  
    --input-storage-location Bucket=<bucket-name>,Key=<path-to-tarball>  
     
   NOTE: <path-to-tarball> is <dcp-folder-name>/<tar-file-name>
9. You will now receive an AFI ID and AGFI ID. The AGFI ID will be what you use to load to the FPGA. Before being able to load the AFI onto the FPGA, you will need to wait until its status goes from “pending” to “available”. To check its status, use the command below using the AFI ID.

$ aws ec2 descript-fpga-images –fpga-image-ids <AFI-ID>

**Section VI: Creating Your Own Custom Logic Using the User App Structure**

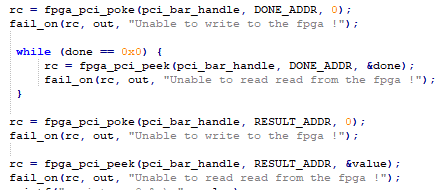
1. Using the provided user app code, in user\_app.vhd replace the fib entity with one you want to use and connect it with the memory\_map entity.
2. In memory\_map.vhd modify the I/O signals and addressing to desired.
3. Connect to and set up your FPGA Developer AMI instance (see Section V steps 1 - 5).
4. Use WinSCP to replace hello\_world.sv in the design folder with the one provided and add the rest of your files in this folder.
5. Using WinSCP, modify the encrypt script in the build/scripts folder to include the new/modified files you have added. Example of lines of code added shown below:

file copy -force $CL\_DIR/design/user\_app.vhd $TARGET\_DIR

file copy -force $CL\_DIR/design/memory\_map.vhd $TARGET\_DIR

file copy -force $CL\_DIR/design/fib.vhd $TARGET\_DIR

1. Now, you can build and run the AFI as normal (see Section V steps 6 - 9 and Section IV).
2. When running the AFI, the test\_hello\_world C code in the software/runtime folder needs to be modified. Inside the peek\_poke\_example function, change the values and addresses that you want to read and write to. fpga\_pci\_poke allows you to write to an address and fpga\_pci\_peek allows you to read from an address. Due to some limitations of the current user app structure, before using fpga\_pci\_peek on an address, you must use fpga\_pci\_poke on that same address prior. An example is show below.



DMA:  
Need to first install edma drivers:  
SSH into instance using “centos”.  
sudo yum groupinstall “Development Tools”

Sudo yum install kernel-devel

Clone github

Go to edma folder

Make

Sudo insmod edma-drv.ko (installs the driver)

lsmod | grep edma (checks if driver is installed)