GeneSys

AWS FPGA Guide

ACT Lab

1 Prerequisites

To build and run an FPGA design on AWS, you must have access to an SSH client and an AWS account.

1.1 Creating a CX Instance

1. Start by creating a CX4Large instance, which will be used to build Genesys design. To do this, go to the AWS dashboard and launch an instance with the following features:
   a. AMI - FPGA Developer AMI
   b. Instance - c4.4xlarge
   c. Storage - Set to at least 128GB
   d. All other settings should be fine as the default selection
2. Generate a .pem file (select "Create a new pair key") and download it into the directory you will be sshing into the instance from.
3. Change the permissions of the .pem file:
   a. `$ chmod 400 *.pem`
4. To connect to the instance, follow the commands in the "connect" section of the EC2 dashboard.

1.2 Configuring CX Instance

1. Generate a set of security credentials in your AWS console. To do so follow this link: https://console.aws.amazon.com/iam/home?#/security_credential
2. Run the following command:
   ```
   $ aws configure
   ```
3. Determine a bucket-name, dcp folder name, and log folder name. Now, run the following commands to create the directories:

```
$ aws s3 mb s3://<bucket-name> --region us-east-1
$ aws s3 mb s3://<bucket-name>/<dcp-folder-name>
$ touch FILES_NAME.txt
$ aws s3 cp FILES_NAME.txt s3://<bucket-name>/<dcp-folder-name>/
$ 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 establishes your bucket name. The second command creates your design checkpoint folder. The third command creates a blank file and then we put this file into the dcp folder. This step is required to create your dcp folder. The fifth command generates your log folder, and finally you place a .txt file in this directory to ensure the folders to be created.

1.3 Setting up AWS FPGA Directories on CX Instance

1. Clone AWS's FPGA repo and run the required HDK and SDK setup scripts.
   ```
   $ git clone https://github.com/aws/aws-fpga
   $ cd aws-fpga/
   ```

```
$ source hdk_setup.sh
$ source sdk_setup.sh
$ source vitis_setup.sh
```

2. In addition, you have to move the FPGA platform folder to the local Vitis directory:

```
cd /home/centos/aws-fpga/Vitis/aws_platform/

sudo cp -r /home/centos/aws-fpga/Vitis/aws_platform/xilinx_aws-vu9p-f1_shell-v04261818_201920_3
/opt/Xilinx/Vitis/2021.2/platforms
```

3. You can now experiment with the example designs located in hdk/cl/examples/

1.4 Creating a F1 Instance

1. There is a chance you do not automatically have access to an F1 instance. To request access you will have to request a limit increase. Once approved please use the following settings:
   a. AMI - FPGA Developer AMI
   b. Instance - f1.2xlarge
   c. Storage - Set to at least 128GB
   d. All other settings should be fine as the default selection
2. You should be able to use the same .pem file as the one created with the cx instance.

1.5 Setting up AWS FPGA Directories on F1 Instance

1. Clone AWS's FPGA repo and run the required HDK and SDK setup scripts.

```
$ git clone https://github.com/aws/aws-fpga
$ cd aws-fpga/
$ source sdk_setup.sh
$ aws configure
```

2. You can now experiment with the example designs located in hdk/cl/examples/

2 Running GeneSys Compiler on AWS

1. Clone the codelet compiler directory:

```
$ git clone https://github.com/actlab-genesys/GeneSys.codelets.git
$ cd GeneSys.codelets
```

2. Create a Python virtualenv:

```
$ python -m venv .venv
$ source .venv/bin/activate
$ python -m pip install pip --upgrade
```

3. If you already have a working installation of Python 3.8, the easiest way to install the GeneSys Codelet compiler is:

```
$ pip install .
```

4. Compile DNNs to GeneSys. The compiler has two entry points that can be accessed from the command line after building:

```
$ compile-genesys -m <model_path> [options]
```

Required Arguments:

-m or --model: Path to the ONNX file containing the DNN model to compile.

3 Building GeneSys on AWS EC2 Instance

3.1 Building Genesys Hardware Binary

1. Proceed to aws-fpga/hdl/cl/developer_designs. Clone the genesys directory within the developer_designs folder:

```
$ cd /home/centos/aws-fpga/hdk/cl/developer_designs/
$ git clone https://github.com/actlab-genesys/GeneSys.git
```

2. Go to aws-fpga/vitis/platforms. Within this folder, you should see what F1 FPGA is available to you. This should match the platform and part selections inside the Genesys directory.

```
$ cd aws-fpga/Vitis/aws_platform/
$ ls
```

   a. Example Platform - xilinx_aws-vu9p-f1_shell-v04261818_201920_3
   b. Example Part - xcvup9p-flgb2104-2-i

Below are locations where the platform and part number are used.

   a. genesys-fpga/fpga_framework/Makefile
      i. Line 36, 39
   b. genesys-fpga/fpga_framework/pack_kernel.tcl
      i. Line 6, 7, 8
   c. genesys-fpga/fpga_framework/systolic_fpga.cfg
      i. Line 1

In addition, edit line 42 of genesys-fpga/fpga_framework/Makefile to be hw

3. Execute the following commands in the fpga_framework directory to build the design:

```
$ make clean
$ make pack_kernel
$ tmux new -s syn
$ make build_hw
```

The output of this process should be a .xclbin file.

3.2 Building Genesys Hardware Emulation Binary

1. For generating FPGA Emulation binary, you will need to complete steps 1 and 2 in Section 3.1.
2. Edit line 42 genesys-fpga/fpga_framework/Makefile to be hw_emu.
3. Run the following commands:

```
$ make pack_kernel
$ export XCL_EMULATION_MODE=hw_emu
$ make build_hw
```

4 Generating awsxclbin file for F1 Instance

To use the FPGA available on an F1 instance:

1. Proceed to the directory where the xclbin file was generated from the build_hw command. Note the bucket name, dcp folder name, and logs folder name you set in *Section 1*.
2. Execute the following command:

```
$ $VITIS_DIR/tools/create_vitis_afi.sh -xclbin=<input_xilinx_fpga_binary_xclbin_filename>
        -o=<output_aws_fpga_binary_awsxclbin_filename_root> \
        -s3_bucket=<bucket-name> -s3_dcp_key=<dcp-folder-name>   -s3_logs_key=<logs-folder-name>
```

This command will generate the awsxclbin; however, you must wait for amazon's approval as well. To check on this process identify the *_afi_id.txt file. Open this file and identify your afi-id.

3. Next run the following command:

```
$ aws ec2 describe-fpga-images --fpga-image-ids <AFI ID>
```

This will show you your status, and it will also note any errors that occurred during execution. Once the status is designated as "available" you are ready to copy this over to your f1 instance.

5 Generating Software Executable

1. Place the test/instruction directory you wish to execute outside of the genesys directory. If you have used the compiler to generate your instructions, this is what you will need to move.
2. Proceed to the following folder: genesys-fpga/fpga_framework/host. Inside of this folder, find the genesys_driver_b2b.h file, and search for the variable REPO_PATH and set it to where the test directory is.
3. Open the genesys_driver_b2b.cpp. Here, you will find the variable binaryFile. Replace the string "*.xclbin", with the awsxclbin filename.
4. From the fpga_framework directory, run the following command:

```
$ mkdir <bin_folder_name>
$ make build_sw TARGET_DIR=<bin_folder_name> TEST_NAME=<name_of_test>
```

You will need to move the generated executable to the F1 instance when you are ready to run on the FPGA.

6 RTL Simulation in Xilinx Environment

1. Clone the genesys repo
2. Choose a configuration for your test. For example, if you want to run a test on a 16x16 systolic array, edit genesys_systolic/source/config.vh to ensure that ARRAY_N and ARRAY_M are set to 16. Also make sure that the compiler is configured for a 16x16 systolic array. Once you have compiled the instructions.
3. Open genesys_systolic/testbench/generic_tb_files/systolic_fpga_benchmark_config.vh and add an entry with the respective instruction, input, and output files generated from the compiler. You could use one of the example entries too for sanity check or as an example. Ensure you use absolute paths to avoid errors. Ensure valid paths are given for all the file variables as shown in the template even if it is not applicable to your test. For example, ADD_ONLY test does not need a bias input, nevertheless a valid path is given for the variable.
4. Start Vivado and add all the files in the subdirectories of genesys_systolic/source/ and genesys_systolic/testbench/generic_tb_files/ as sources.
5. You will need to generate the AXI Verification IPs for running simulation. In Vivado, go to IP Catalog and look for AXI Verification IP. Six AXI VIPs need to be created and use the same names as below. This might require a Xilinx Vivado License.

7 Running Genesys on a test

7.1 Running on F1 Instance using Software Executable

1. SSH into your F1 instance. Clone the aws-fpga repo here:
   ```
   $ git clone https://github.com/aws/aws-fpga.git $AWS_FPGA_REPO_DIR
   ```
2. Run the runtime setup script in the aws_fpga directory:
   ```
   $ source vitis_runtime_setup.sh
   ```
3. Finally, make sure that your executable and awsxclbin file are located in this directory, and you can then run the following:
   ```
   ./<executable> <awsxclbinfile>
   ```
   Example: `./FPGA_resnet50_gemm57 aws_systolic_fpga.awsxclbin`

   If using one of the example tests, if the test passes, then "TEST PASSED" will be displayed.

7.2 Running on F1 Instance using Python Driver

If you use the python driver available under fpga_framework/host_py/host.py , then you do not ned to complete Section 5. This driver's inputs are your test/instructions and awsxclbin file.

1. In fpga_framework/host_py/host.py, set the following paths - test_name, data_info_file, base_path, genesys_binary
2. Run the following command:
   ```
   $ python3 host.py
   ```

8 DC Synthesis

1. Move read_rtl_genesys.tcl and run_dc_SIMD_4x4.tcl to a directory containing the needed rtl files which are listed in *read_rtl_genesys.tcl*
2. Run $ dc_compiler to launch the compilation tool
3. Finally, run $ source run_dc_genesys.tcl

9 Additional

9.1 Using X11 on AWS

1. You can access all Vivado and Vitis tools on AWS. To access the gui using x11, you will first have to set your DISPLAY variable from your terminal. When you ssh into AWS, make sure you include the -X flag.
2. Run the following command in AWS:

```
$ sudo  yum groupinstall "X Window System"
```

You will probably need to exit and re-enter the instance after running this command.

To launch vivado, all you should need to do is run:

```
$ vivado
```

For further assistance, please refer to the following aws guide on using x11 forwarding.

https://aws.amazon.com/blogs/compute/how-to-enable-x11-forwarding-from-red-hat-enterprise-linux-rhel-amazon-linux-suse-linux-ubuntu-server-to-support-gui-based-installations-from-amazon-ec2/

9.2 Resources

- Running a Design on F1 Instance:
  - **https://github.com/aws/aws-fpga/tree/master/Vitis#hw**
- AWS Setup and running example program:
  - https://caslab.csl.yale.edu/courses/EENG428/19-20a/tutorials/getting_started_hello_world.pdf