**A step-by-step walkthrough for On-Premise Develop and F1 Deploy**

In order to develop any SDAccel Application on-premise, the user needs proper SDAccel Software version and license. Please refer to the [Github Page for Software download and License instruction.](https://github.com/aws/aws-fpga/blob/master/hdk/docs/on_premise_licensing_help.md)

The supported Operating Systems for SDaccel On-premise development are:

·         Red Hat Enterprise Workstation/Server 7.2 and 7.3 (64-bit)

·         Red Hat Enterprise Workstation 6.7 and 6.8 (64-bit)

·         CentOS 6.8, CentOS 7.3 (64-bit)

·         Ubuntu Linux 16.04.1 LTS (64-bit)

**Step 1: Downloading the SDx Development tool**

SDAccel software download link: [SDAccel Download Link](https://www.xilinx.com/member/forms/download/xef.html?filename=Xilinx_SDx_op_2017.1_sdx_0715_1_Lin64.bin&akdm=0)

MD5 SUM Value: cfa96d3af608954eb2bc0bf4abb892f8

**Step 2: License and tool installation**

New users will need to obtain an on-premises license of Vivado. Follow instruction to request for a node-locked or floating license: <https://www.xilinx.com/products/design-tools/acceleration-zone/ef-vivado-sdx-vu9p-op-fl-nl.html>

(If required, for additional details for SDx installation, you may refer to [Sdx Installation and Licensing guide](https://www.xilinx.com/html_docs/xilinx2017_1/sdaccel_doc/topics/introduction/concept-supported-boards-release-notes.html)).

**Step 3: Download the aws-fpga Git repository in Local Host.**

We will use Vector Addition with RTL Kernel example from SDAccel Github to demonstrate on-premise development procedure. First, we will download the Git repository for this purpose.

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

**Step 4: Download the Platform and example directories.**

|  |
| --- |
| $ cd aws-fpga-preview  $ source sdaccel\_setup.sh |

Sourcing sdaccel\_setup.sh will show some errors as it also tries to install runtime driver that requires sudo access. However, the errors are nonintrusive, and the user can ignore those error messages. In the end, this process will download hardware platform and Github examples.

**Step 5: Hardware build for RTL Kernel example:**

|  |
| --- |
| cd sdk/SDAccel/examples/xilinx/getting\_started/rtl\_kernel/rtl\_vadd  make all TARGETS=hw DEVICES=$AWS\_PLATFORM |

The build process will generate the Host and Kernel executables.

Host executable: host

Kernel executable: ./xclbin/vadd.hw.xilinx\_aws-vu9p-f1\_4ddr-xpr-2pr\_4\_0.xclbin

**Step 6: Transferring Host and Kernel executable on F1 instance**

Next step is to transfer both the host and kernel executables on the F1 instance.

To upload data to the F1 instance, use scp with the same elements as the SSH login:

|  |
| --- |
| scp -i ~/<pem-name>.pem <from file> <login-id>@<host-ip>:/home/<login-id>/<to file> |

**Step 7: Login to the F1 instance**

Here we assumed you already setup your AMI with CLI, S3 bucket and runtime driver. If not, please skip the rest of this guide and follow the [README.md](https://github.com/aws/aws-fpga-preview/blob/develop/sdk/SDAccel/README.md) file (from the section "Create an Amazon FPGA Image (AFI) for your kernel") for the remaining instructions.

However, if your AMI is already configured with S3 bucket, runtime driver and CLI, then you can follow next three simple steps procedure.

**Step 8: Creates an AWS FPGA binary file with an AWSXCLBIN extension**

Use the script create\_sdaccel\_afi.sh to convert the .xclbin file (specified by –xclbin switch) to .awsxclbin file (specified by –o switch) as shown below. You also need to provide the location of S3 bucket, dcp-folder and log-folder.

|  |
| --- |
| SDACCEL\_DIR/tools/create\_sdaccel\_afi.sh \  -xclbin= vadd.hw.xilinx\_aws-vu9p-f1\_4ddr-xpr-2pr\_4\_0.xclbin \  -o=<output\_awsxclbin\_filename> \  -s3\_bucket=<bucket-name> \  -s3\_dcp\_key=<dcp-folder-name> \  -s3\_log <logs-folder-name> |

The above step will generate <>.awsxclbin file (Specified by -o switch). You will also get an AFI (Amazon FPGA Image) ID file containing AFI ID of the process. The AFI ID can be used to check the completion of the AFI generation process (as shown in Step 9c below).

**Step 9: Prepare the executable file**

a) Rename the .awsxclbin to original .xclbin file.

|  |
| --- |
| mv <generated>.awsxclbin vadd.hw.xilinx\_aws-vu9p-f1\_4ddr-xpr-2pr\_4\_0.xclbin |

b) Prepare a test.sh to execute the host executable.

|  |
| --- |
| source /opt/Xilinx/SDx/2017.1.rte/setup.sh  ./host |

c) Check Status of AFI generation process using AFI-ID

|  |
| --- |
| cat <timestamp>\_afi\_id.txt # Note the AFI-ID from this command  #Checking the status  aws ec2 describe-fpga-images --fpga-image-ids <AFI ID> |

A finished FPGA image creation job will show "Available" State

|  |
| --- |
| "State: {  "Code" : Available"  } |

**Step 10: Once available execute the application on AWS F1 instance**

|  |
| --- |
| [centos@ip-172-31-79-92]$ sudo ./test.sh  Device/Slot[0] (/dev/xdma0, 0:0:1d.0)  xclProbe found 1 FPGA slots with XDMA driver running  INFO: Importing ./vadd.hw.xilinx\_aws-vu9p-f1\_4ddr-xpr-2pr\_4\_0.xclbin  INFO: Loaded file  INFO: Created Binary  INFO: Built Program  All Device results match CPU results!  TEST PASSED. |