**Lab 5: Implementation of AES with RTL (VHDL)**

**Goal:**

In this lab, students will learn how to convert a software solution to a hardware using standard RTL and HDL programming. Moreover, they will implement the accelerator on PAC.

**Steps overview**:

1. Read AES\_RTL.pptx guideline
2. VHDL cheat sheet is provided
3. Learn how to use Intel Quartus (tutorial\_quartus.pptx)
4. Design AES in VHDL based on the specification provided in the guideline
5. Simulate your design using Intel Quartus
6. Implement the AES for PAC and run on devcloud

Step 4:

Please develop the RTL code of AES. Based on the RTL design, you must design the Datapath and the controller unit for your hardware. Then, perform the implementation of the design. In addition, students learn how to understand and interpret the reports generated by hardware simulation tool for FPGA hardware profiling.

Step 6:

1. AES\_ccip\_mmio has been provided. It contains two folders: /hw, /sw
2. Put your design vhdl file into /hw directory. In the hw directory, a systemverilog wrapper is provided for you.
3. There is filelist.txt in /hw directory update it with your vhdl file names.
4. Login to devcloud using ssh (instruction has been provided in Lab 1)
5. Copy AES\_ccip\_mmio to your devcloud space (using scp instruction discussed in lab 1)
6. When you are logged in, run following commands:

* source /data/intel\_fpga/devcloudLoginToolSetup.sh
* devcloud\_login

1. then select 1 (Aria 10 compilation and run) and then select 0 (Aria 10 sdk 1.2).
2. Now you are logged in on new node that has capability of working with FPGA! On this new node, run following commands:

* source /data/intel\_fpga/devcloudLoginToolSetup.sh
* tools\_setup -t QP 19.2
* tools\_setup -t A10DS
* echo $OPAE\_PLATFORM\_ROOT
* which quartus

1. Now you should see no error.
2. In AES\_ccip\_mmio directory, run following commands:

* afu\_synth\_setup --source hw/filelist.txt synth
* cd synth
* ${OPAE\_PLATFORM\_ROOT}/bin/run.sh

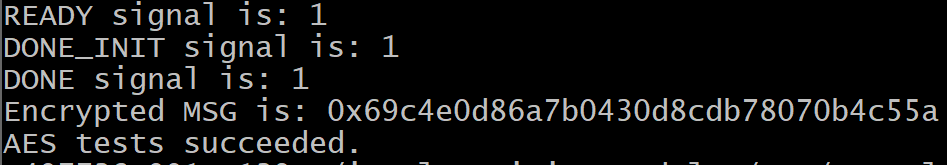
1. In this step, quartus will run and perform the synthesize. If there is any error, remove /synth directory and after modifying your vhdl codes, redo step 10.
2. Go to synth/build/output\_files/timing\_report/ to check timing report.
3. clocks.sta.fail.summary file must be empty.
4. If everything goes well, you will see afu.gbs file in /synth directory.
5. Run following command to load bitstream into FPGA:

* fpgaconf -B 0x3b afu.gbs

1. After loading is done, go to /sw directory.
2. Run following commands:

* make clean
* make
* ./afu

1. If your design works based on the specification, you should see the following output:



If you are interested in learning more about core cache interface protocol and memory mapped I/O, and how the interface works, please watch the following tutorials:

1. Intel PAC overview:

<https://www.youtube.com/watch?v=HatHuLtZ5-0&feature=youtu.be&ab_channel=GregStitt>

1. CCI-P Explanation:

<https://www.youtube.com/watch?v=e03xuTsQ4fQ&ab_channel=GregStitt>

1. ccip\_mmio RTL Code Demonstration:

<https://www.youtube.com/watch?v=3WXo1qzYTvs&ab_channel=GregStitt>

1. ccip\_mmio SW Code Demonstration:

<https://www.youtube.com/watch?v=Qed4ooAeepw&ab_channel=GregStitt>

1. Synthesizing AFU and Configuring PAC on the DevCloud:

<https://www.youtube.com/watch?v=QPjkVo3gSb0&feature=youtu.be&ab_channel=GregStitt>