Assignment 1
Design and Synthesis of
Advanced Encryption Standard Algorithm
Due date: 02/17/2020 (Tuesday)

Description:

In this assignment, you will be implementing the pipelined and iterative architectures of the advanced encryption standard (AES-128) algorithm (**encryption only**, including the key scheduler) using Verilog/VHDL. These implementations should be synthesized using Xilinx Vivado software (FPGA design flow) to determine the area and power consumed by each architecture. You should also compare and contrast the following five metrics: throughput, latency, area, power, and delay for each of the architectures. **For extra credits,** you can synthesis the same designs using Synopsys design compiler (DC) (ASIC design flow).

For assignment 1, you are expected to:

i. Implement the AES-128 algorithm (**encryption only**, including the key scheduler) using Verilog or VHDL. Please refer to the resources section for example codes and learning aids.

ii. Test and verify your RTL code using the Cadence NC-Verilog simulator (or you can use any simulator which you are comfortable with).

iii. Synthesize the verified RTL using Xilinx Vivado software. Report the area and power consumed for each architecture.

iv. Determine the throughput, latency, and delay for each architecture.

v. Share your inference on these five metrics.

Extra credit:

i. Synthesize the verified RTL using Synopsys DC – (ASIC design flow) and report the area and power for both the architectures.

Understanding the metrics:

1. **Throughput:** The number of encrypted text produced every second.
2. **Latency**: The number of clock cycles required to perform one encryption.
3. **Area:** Area consumed by the architectures.
4. **Power**: Power consumed by the architectures.
5. **Delay**: The maximum clock frequency (minimum time period) required by your design to run without any timing errors (no setup/hold slacks). This clock frequency will be reported by the Vivado in the timing report.

Resources required:

1. **Understanding the Algorithm:**
   a. Class Notes.
   b. AES documentation from NIST: https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.197.pdf (Uploaded in eCampus along with the manual)
   c. A python version of the AES algorithm is uploaded in the eCampus for your reference (aes.py). Execute this file using *python aes.py* in a Linux machine or bash terminal where python is installed. Apollo/Hera servers have python installed in them. You can always check the ciphertext for other plaintext/key pairs by modifying line numbers 667 and 668. Key for encrypting different plaintexts should be the same.
   d. Additional materials:
      i. https://www.comparitech.com/blog/information-security/what-is-aes-encryption/
      ii. https://www.youtube.com/watch?v=NHuibtoL_qk

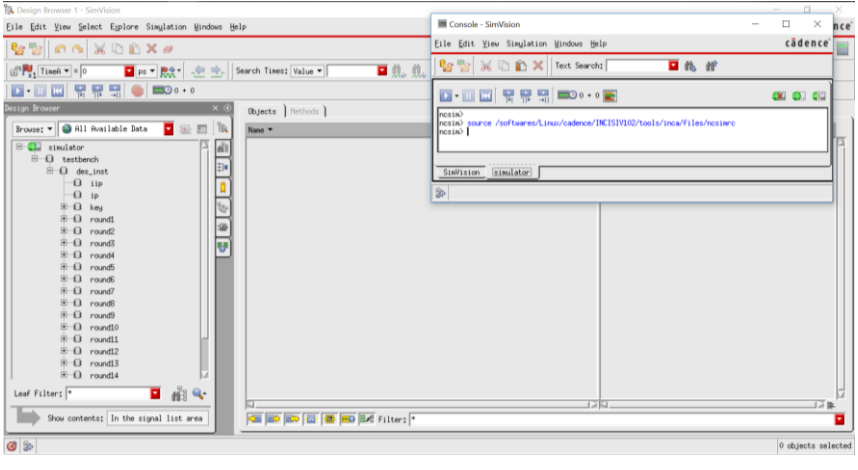
2. **Verilog/VHDL Implementation:** You can use either Verilog/VHDL language for this implementation in your preferred text editors (Vim, Emacs, gedit, pico, etc.). Few web references:
   a. http://www.asic-world.com/verilog/veritut.html
   b. http://www.asic-world.com/vhdl/tutorial.html
   c. http://www.sunburst-design.com/papers/ (One of the best websites to learn RTL designing). Look out for SNUG Verilog papers (1998 to 2003).
   d. Verilog Quick reference guide is uploaded in the eCampus.

3. **NC-Verilog Setup:**
   a. Execute the following command in the bash terminal to source the cadence to use NC-Verilog simulator:
   source /softwares/setup/cadence/setup.incisive152.bash
   b. Run the command:
   ncverilog +access+r +gui ***your_testbench.v your_module.v.***
   e.g.: ncverilog +access+r -gui -sv testbench_aes_128.v aes_128.v

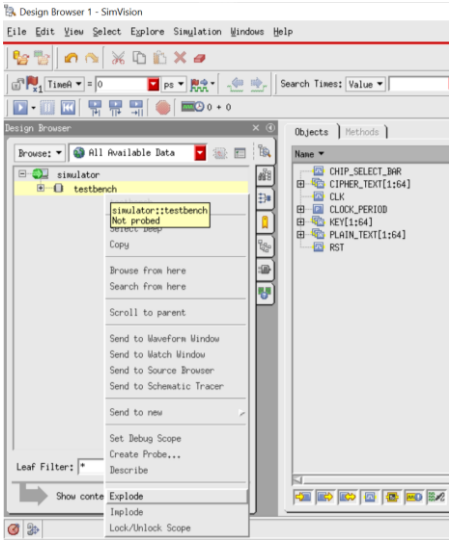
      i. ***your_testbench.v*** → name of your test bench file. In this file, you will be instantiating the AES top module. The plaintext and key will be sent to the AES module and the ciphertext will be received from the module.
      ii. ***your_module.v*** → name of your Verilog design file of AES encryption top-level file. **`*include*** all the other Verilog/VHDL sub-module files in the top-level file (aes_128.v) as shown in the below figure. Otherwise, you have to specify all the RTL files used in the above command.

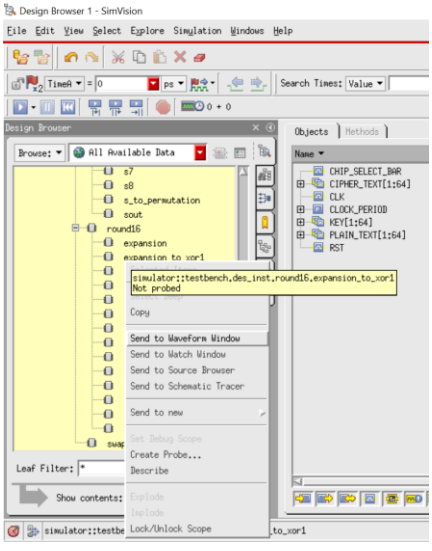
```
/*
 * Copyright 2012, Homer Hsing <homer.hsing@gmail.com>
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * you may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 * http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 */
`include "/home/grads/g/gjn/design_compiler/synthesis/aes/rtl/round.v"
`include "/home/grads/g/gjn/design_compiler/synthesis/aes/rtl/table.v"

module aes_128(clk, state, key, out);
    input         clk;
    input  [127:0] state, key;
    output [127:0] out;
    reg    [127:0] s0, k0;
    wire   [127:0] s1, s2, s3, s4, s5, s6, s7, s8, s9,
                   k1, k2, k3, k4, k5, k6, k7, k8, k9,
                   k0b, k1b, k2b, k3b, k4b, k5b, k6b, k7b, k8b, k9b;

    always @ (posedge clk)
      begin
        s0 <= state ^ key;
        k0 <= key;
      end
```

c. Executing this command opens the **Design browser** and **Console** windows:

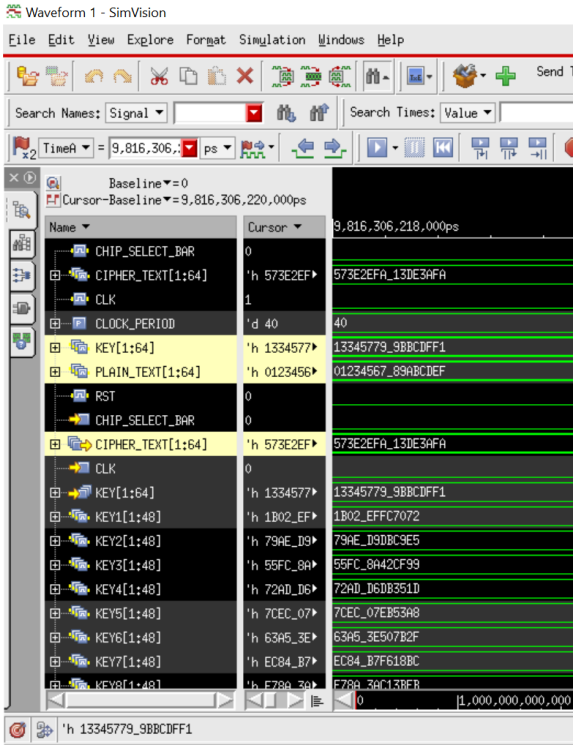


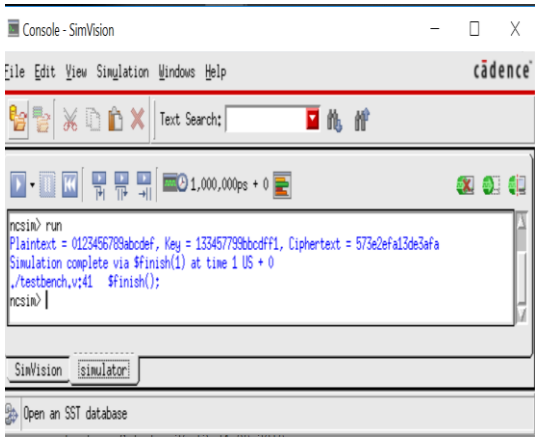
d. To view the signals in the waveform window after running the simulations, right-click on the top-level testbench module and click on **Explode**. Select all the modules in the Design browser, right-click and select **Send to waveform window**.





e. Run the simulation by clicking on **Simulation → Run** in the **Console** window. You can view the output in the waveform window and printed logs (if you are printing anything in the testbench file using $display/$monitor statements) in the **Console.** You can either share the screen capture of the **waveform window** or the **Console** log showing the plaintext and key inputs and the corresponding ciphertext output. **Note:** It is always easier to report the console log rather than the waveforms. Waveform window is used mainly for **debugging** purposes.

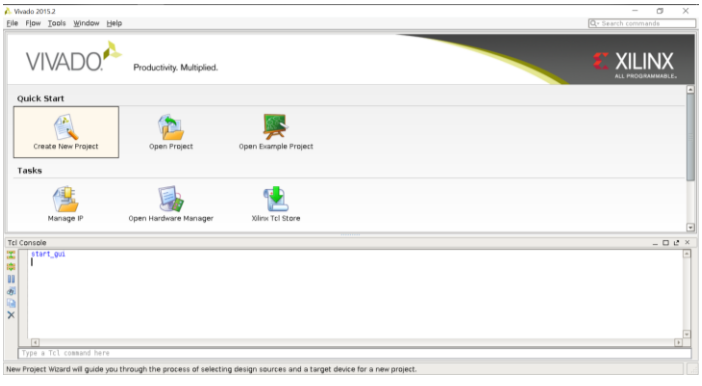




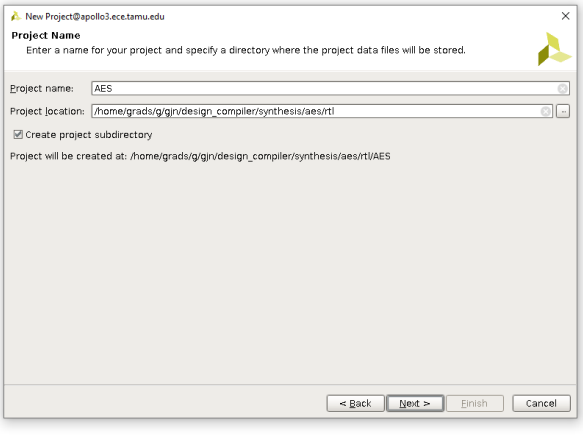
4. FPGA Synthesis:

a. Include the following line in your ~/.bashrc and open a new terminal
source /softwares/Linux/xilinx/Vivado/2015.2/settings64.sh

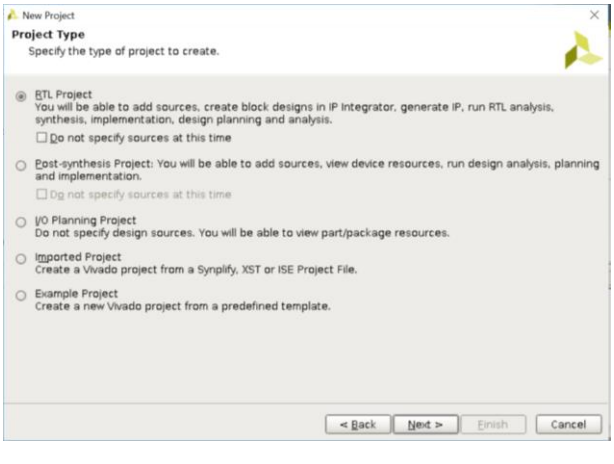
b. Invoke the tool with: *vivado &*



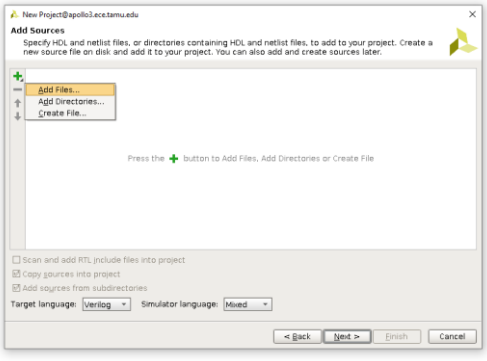
c. Select “Create new project” from the Quick Start menu and provide the project name and path where the project run files are stored.

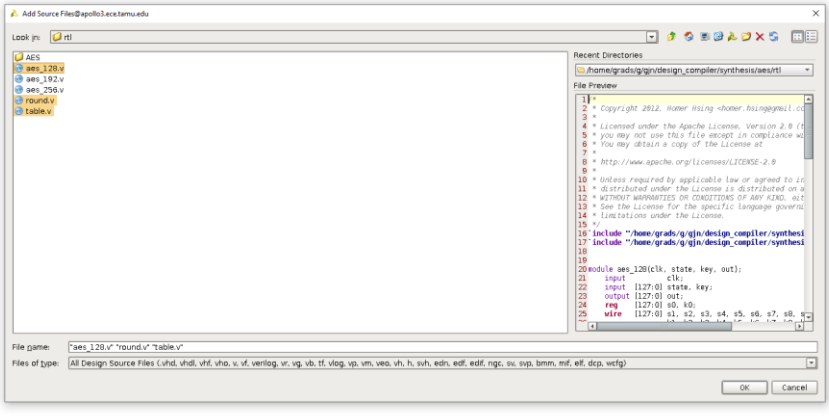


d. Select project type as RTL project, as we will be synthesizing the design using HDL files.



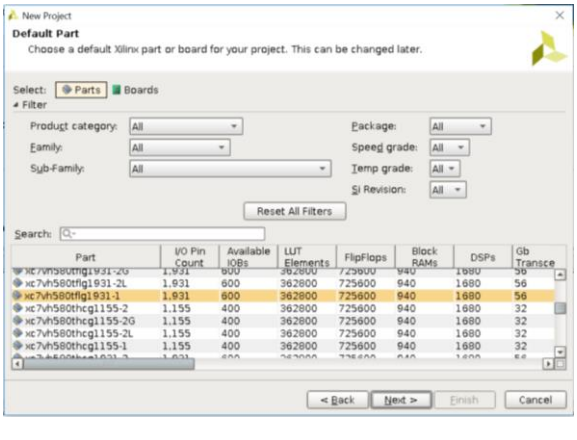
e. Click on the “Add sources → Add files” and select all the RTL only files (do not include testbench related files).



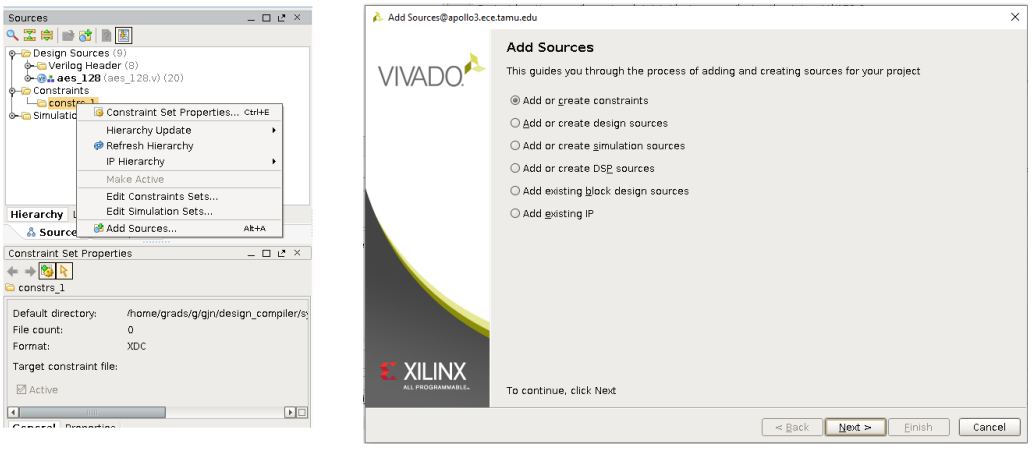


f. As we will not include any intellectual property (IP) or constraints for this synthesis, you could safely skip them in the project creation wizard.

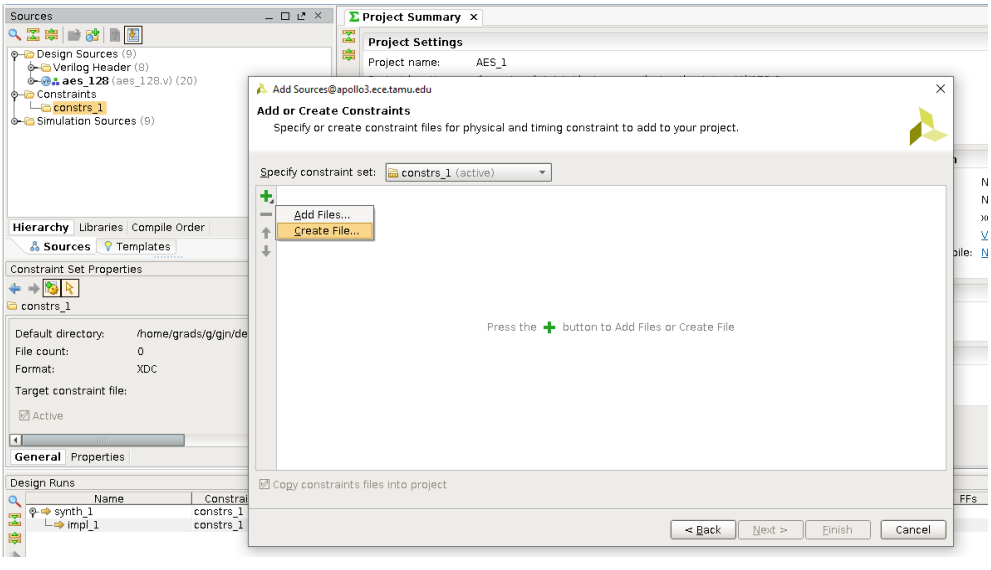
g. Select any one of the available FPGA (part number) from the wizard and click finish.



h. Add clock constraints which is required to determining the maximum clock frequency. In the “Sources” window click on the “Constraints” folder. Right clock on “constrs_1” folder and select “Add sources”. Select the “Add or create constraints” as shown below:

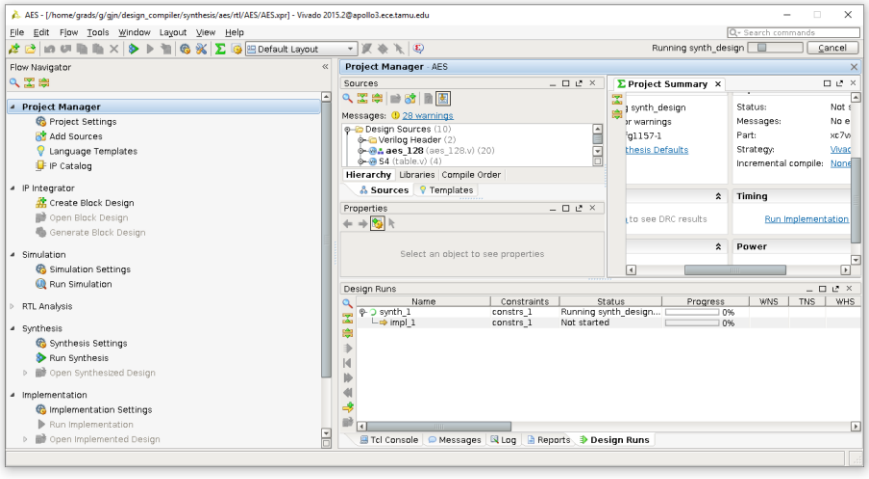


i. Select “Create new file” and name the constraints file as shown below:

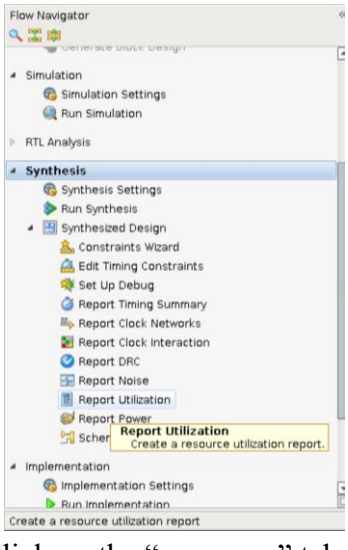


j. Once the empty constraints file is generated, include the below clock constraints:
create_clock -name clk_n -period 10.000 -waveform {0 5} [get_ports clk]
The above command assigns a clock constraint to the clock port named by “clk” in the AES top.v file. The period of the clock is 10nm (100MHz). The duty cycle of the clock is 50% which is specified by the “waveform” switch that the posedge of clock is at 0ns and negedge of the clock is at 5ns.

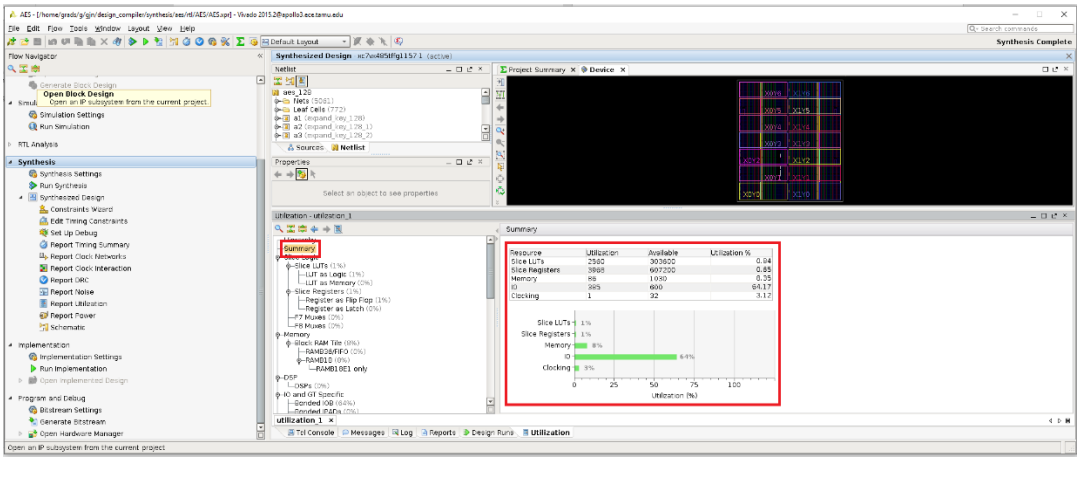
k. Select the top-level file from the **Sources** window and click on **Run Synthesis** from the synthesis tab of the project manager. This process generates the synthesized netlist and also the report having the area utilization details.



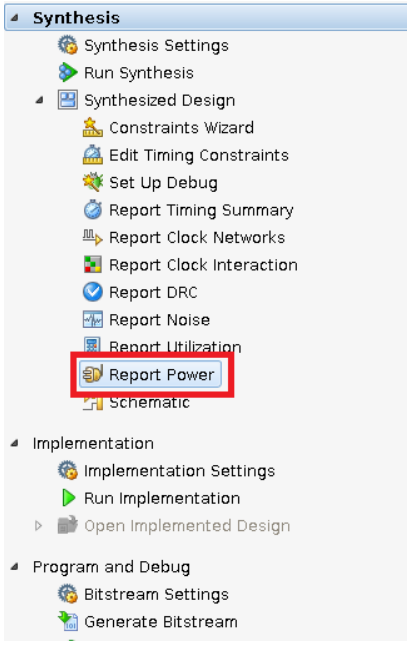
l. Run **Report Utilization** from the synthesis menu, as shown below

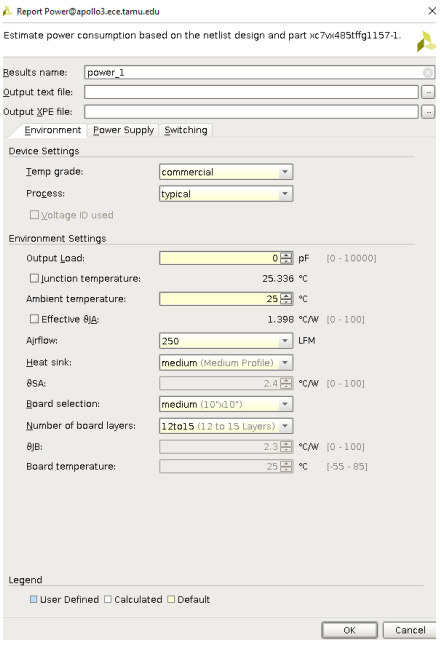


m. Click on the “summary” tab in the utilization window to view the area utilization details, as indicated by the highlighted red box below

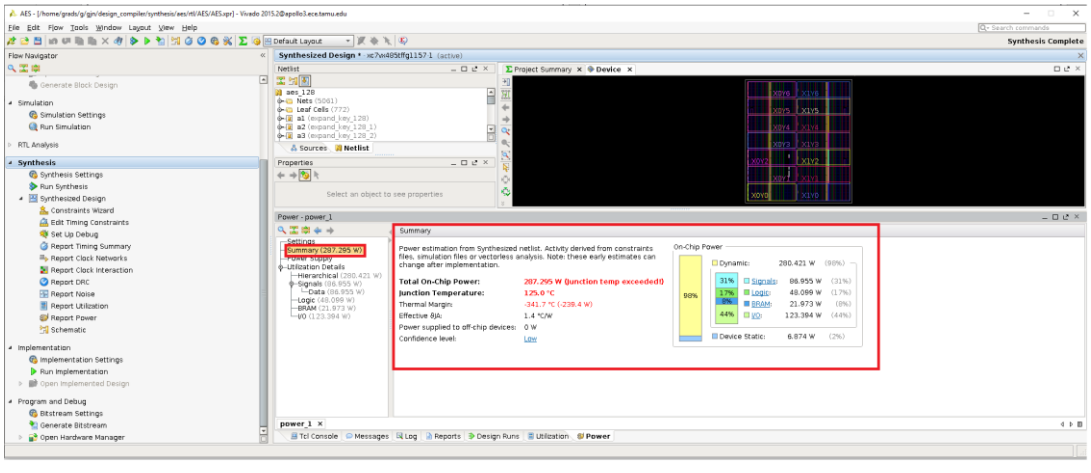


n. Similarly, run **Report Power** from the synthesis menu. As shown below, use the default settings.



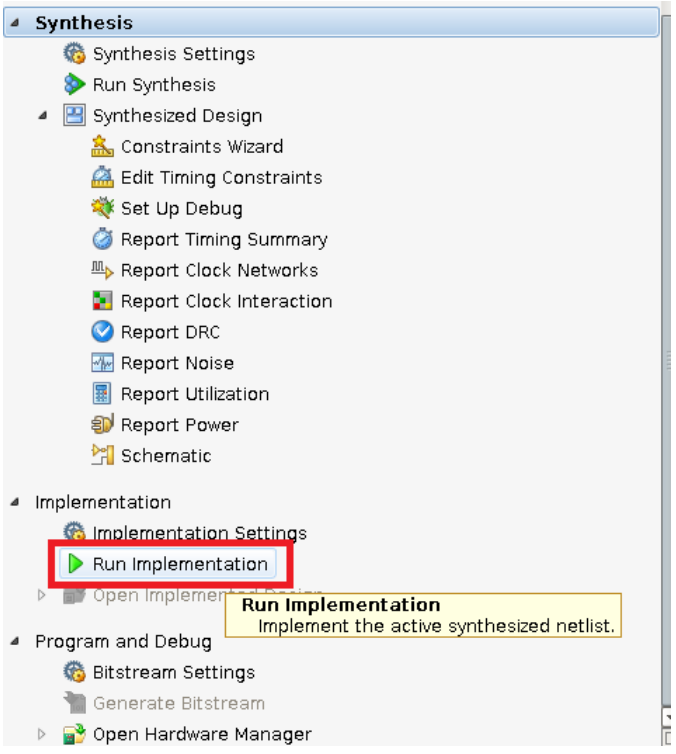


o. Click on the "summary" tab in the utilization window to view the power consumption details, as indicated by the highlighted red box below

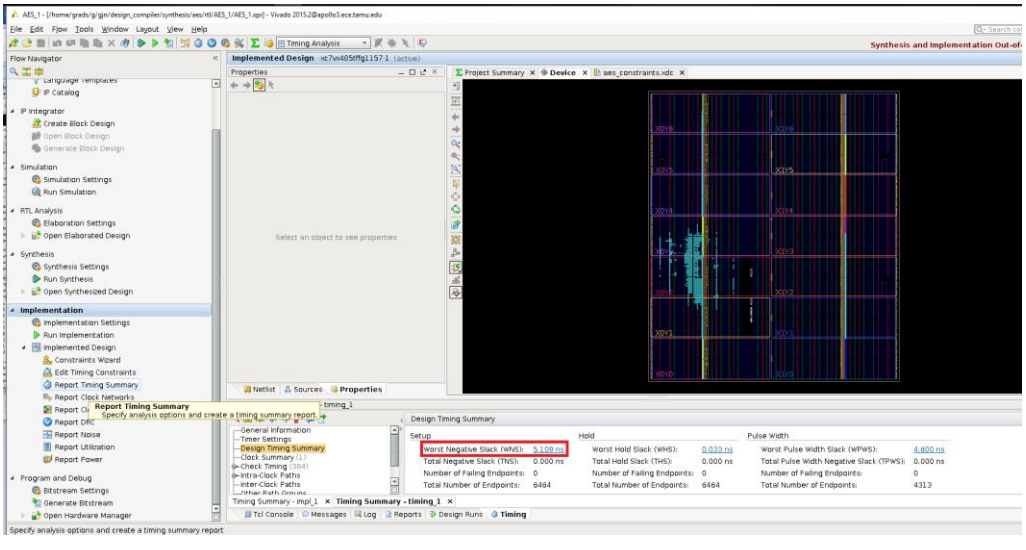


p. Please note that the area and power consumption of the same design vary based on the chosen FPGA.

q. For calculating the maximum clock frequency (or minimum time period), please click "Run Implementation".



r. For calculating the maximum clock frequency (or minimum time period), click on the report timing summary, "Design Timing Summary" reports the worst negative slack (which is the delay taken by the logic elements inside the design). The minimum time period is given by: clock_period (mentioned in constraints file) – worst negative slack (reported in the Design Timing Summary). The reciprocal of the minimum clock period gives the maximum frequency of operation.



For ASIC synthesis – Extra credits:

a. **Synopsys Design Compiler:**
   i. Include the following line in your ~/.cshrc file (if you are using a c or t-shell) and open a new terminal. *$source /softwares/setup/synopsys/setup.synopsys.tcsh*
   ii. Else include the following line in your ~/.bashrc (if using a bash shell) and open a new terminal. *$source /softwares/setup/synopsys/setup.ecen454.bash*
   iii. The sample script "aes_synthesis.txt" is in the lab folder for your reference. Update this script with new file names that you have.
   iv. Place the RTL files and script in the same directory.
   v. Execute *design_vision* in the terminal to invoke the design compiler (DC).
   vi. Execute "*source aes_synthesis.txt*" in DC. This converts the RTL files into the synthesized netlist.
   vii. Report area and power using (a) *report_area -nosplit > log_file_name_area.log* and (b) *report_power -analysis_effort high > log_file_name_power.log* (These commands are already included in the aes_sythesis.txt file)

Due date and Deliverables:

The due date to submit the codes/log files/synthesized netlist files is on **02/17/2020**.

The following deliverables have to be submitted in a zip file format with assignment1_<last_name>_<UIN>.zip as the file name in the eCampus.

1. RTL and Testbench codes (Verilog/VHDL).
2. The report containing the simulation log files that shows the functioning of the AES algorithm. Test the AES-128 with key = **000102030405060708090a0b0c0d0e0f** and plaintext = **00112233445566778899aabbccddeeff** (Ciphertext = **69c4e0d86a7b0430d8cdb78070b4c55a**)**.** For the given key, print the ciphertext demonstrating the encryption. You can also add different keys/plaintext pairs. (Text logs are preferable than screen captures of the simulation window!).
3. Include the simulation command you used for compiling and simulating the code.
4. For FPGA synthesis share the area utilization and power consumption logs for both pipelined and iterative architecture
5. Compare and contrast the five metrics for the pipelined and iterative architecture using the table below.

S. No	Metric	Pipelined architecture	Iterative architecture	Inference (Please explain why the values of the metrics are different between the two architectures)
1	Throughput			
2	Latency			
3	Area			
4	Power			

5	Delay			

6. For ASIC synthesis, share:
   a. Area utilization log from Synopsys DC.
   b. Power consumption log from Synopsys DC.
   c. Synthesized Verilog netlist of AES-128 implementation.
7. **Please ensure that the timestamp and your netid are included in all the tool generated log files.**

Lab 1 Rubrics:

1. Simulation logs, area utilization log, and power consumption logs – 5 points
2. Comparison of the five metrics between the pipelined and iterative architecture – 5 points
3. Area and power utilization logs and synthesized netlist file from ASIC synthesis – 2.5 points **(Extra credit)**