ECE 3170 Cryptographic Hardware for Embedded Systems

Fall 2025

Assoc. Prof. Vincent John Mooney III

Georgia Institute of Technology

Lab 3, 100 pts.

Due Tuesday, October 21st prior to 11:55pm

(please turn in homework electronically on Canvas)

Implementing Triple DES

In this lab, you will modify DES from Lab 1 to perform Triple DES. Detailed information about Triple DES can be found here and here. We will be using keying option 1 for Triple DES. This means that a single Triple DES encryption requires three standard length DES keys. For the rest of this lab, we will refer to 3 keys as 1 set.

The encryption algorithm is as shown below:

$$ciphertext = E_{K3}\left(D_{K2}\left(E_{K1}(plaintext)\right)\right)$$

The decryption algorithm is as shown below:

$$plaintext = D_{K1}\left(E_{K2}\left(D_{K3}(ciphertext)\right)\right)$$

The basic encryption and decryption functions above are the same as normal DES, implying Triple DES is essentially just conducting the DES algorithm three times in a row.

I. Generation of Baseline Test Cases using C Code

As with Lab1, you are required to select your own keys. It is advisable to reuse the keys you have tested in Lab 1 and then add in 10 more unique keys to make a total of 15 keys, for 5 sets of keys.

Same as Lab 1, you are required to submit 5 plaintext test cases, 5 ciphertext test cases, and 5 sets. Thus, you will test all 10 input cases on each set for a total of 50 tests (half testing encryption and half testing decryption). The set of keys may not be any of the known weak cases for DES (Please see this for more information). These test cases must be your own (under the GT Honor Code, you are **not** allowed to share your test cases with others) the reason for this is that the focus should be on good coding and testing practices, not comparison with others. However, during Graduate Teaching Assistant (GTA) office hours you may ask if your test cases appear reasonable, and the GTA may (or may not if the GTA does not know!) point out which test cases are not working. The GTA can also look at your code, inspect your choice of compiler and overall C/C++ environment, etc., to help you debug and test your code.

You must provide a rationale for your chosen plaintext cases. The cases must be somewhat unrelated to each other; e.g., you may not choose numeric sequences (e.g., adding one to each prior case).

The test cases and keys must be submitted in .txt files with following names:

- keys.txt (5 sets)
- plaintextin.txt (5 plaintext test cases)
- ciphertextout1.txt (5 ciphertext from using set1 and the plaintextin.txt)
- ciphertextout2.txt (5 ciphertext from using set2 and the plaintextin.txt)
- ciphertextout3.txt (5 ciphertext from using set3 and the plaintextin.txt)
- ciphertextout4.txt (5 ciphertext from using set4 and the plaintextin.txt)
- ciphertextout5.txt (5 ciphertext from using set5 and the plaintextin.txt)
- ciphertextin.txt (5 ciphertext test cases, please pick one ciphertext output from each of the ciphertextout1.txt … ciphertextout5.txt)
- plaintextout1.txt (5 plaintext from using set1 and the ciphertextin.txt)
- plaintextout2.txt (5 plaintext from using set2 and the ciphertextin.txt)
- plaintextout3.txt (5 plaintext from using set3 and the ciphertextin.txt)
- plaintextout4.txt (5 plaintext from using set4 and the ciphertextin.txt)
- plaintextout5.txt (5 plaintext from using set5 and the ciphertextin.txt)

Obviously, the result of decrypting a ciphertext produced by a particular encryption of an input plaintext should result in the same plaintext output (assuming the same set is used). Similarly, the result of encrypting a plaintext produced by a particular decryption of an input ciphertext should result in the same ciphertext output (assuming the same set is used).

Your task for this section is to modify the "main" function from your Lab 1 files. Please also submit a README to show me how to run your C code. Your C code is not required to write ciphertextout.txt and plaintextout.txt. If you choose not to write the results to a file, your program must print the results in the terminal. Please make sure your print statements to the terminal can be understood by the GTA.

II. ModelSim Tutorial and Functional Simulation of VHDL Code

After generating the baseline results for Triple DES using the C code, you will modify the VHDL code from Lab 1 to implement Triple DES. You should be either modifying the file "des_top.vhd" file, a file above the des_top.vhd file, or creating a new file. Modifying purely the test bench does **not** meet the intent of generating the Triple DES module. The test bench should then be modified to correctly test the new Triple DES module.

Please follow the instructions below to simulate all testcases from Section I. The next step is to generate the waveforms of one encryption and one decryption test case as shown below. Note that after your modification to VHDL, you may end up with too many signals in the waveform such that you cannot fit all the testbench signals in a single waveform. In this case, please export the waveform image multiple times to capture all waveform signals.

Same as in Lab 2, to simulate the code, proceed as follows:

- Once ModelSim is open, create a new project by clicking on: File --> New --> Project...
- Give the project a name, say "des", and click OK.
- After creating the project, we have to add all the VHDL files to it. Do so by clicking on: Add Existing File --> Browse...
- Choose all the VHDL files in the directory. Click Open --> OK.
- Once you are done adding ALL the VHDL files, click Close.
- Now we have to compile the design files. To do so click on: Compile --> Compile All. Check the transcript window to make sure all files were successfully compiled.
- Now we can start the simulation by clicking on: Simulate --> Start Simulation... In the Start Simulation window, make sure you are on the design tab, expand the work library, choose the testbench for this code named: "des_cipher_top_tb", and click OK.
- ModelSim will change view into simulation mode and a couple of other windows show up.
- Our next step is to add some signals of interest to a wave window to monitor their changes as simulation proceeds. Before doing so, let us create a dump file that will be storing the results of our simulation as we perform it. To do so, type the following in the command line of the "Transcript" window:
  vcd file sim_results.vcd
  vcd add des_cipher_top_tb/*
  vcd add des_cipher_top_tb/Inst_des_cipher_top_dut/*
- Now let us add our signals of interest to the wave window to monitor their changes as simulation proceeds. To do so, go to the "sim" window and click on the instance named "des_cipher_top_tb" to add the testbench signals. Next, open the "Objects" window and choose all the signals (inputs, outputs, and internal). Right-click on the selection and click on Add Wave. Check that the signals have been successfully added to the "Wave" window.
- Our final step is to run the simulation for a specific time. For this testbench, running the simulation for 2000 ns should be enough. To do so, type **run 2000 ns** in the command line of the "Transcript" window. Please double check that you are getting the expected ciphertextout and plaintextout. It may be possible that your design needs more than 2000ns, so you may want to increase the simulation time.
- Navigating back to the "Wave" window will now show you the result of the simulation for all the signals that we added. To better read the values, select all the signals and right-click, then change the Radix to Hexadecimal. Also, towards, the bottom left of the signals pane (to the left of where it says "Now"), there is a blue button that has a description of "Toggle leaf names <-> full names" if you hover the mouse over it. Click on that button to show the signal names only without the hierarchy.
- Look through the wave window and try to understand how the signals are changing values with respect to the simulation time. Specifically, look for the output signal that shows the encrypted/decrypted value.
- Now that we have run the simulation, make sure that you set the zoom of the wave window to a full view. To do so, right-click anywhere in the wave window and click on Zoom Full. Export an image of your simulated waveform. To do so, click on File --> Export --> Image... and save it as an image. **Submit the waveforms in the lab report verifying one encryption testcase and one decryption testcase in Section III**. Note that you may need to export multiple images to cover all signals in the waveform.
- Before ending the simulation, open the transcript window and verify that the no reports are generated by the testbench indicating a failure of any of the test cases.
- Finally, to end the simulation and correctly save the results in the VCD file, click on: Simulation --> End Simulation.

III. Quartus Utilization

Finally, compile your Triple DES implementation in Quartus and take a screenshot of the resource utilization.

IV. Implementation Questions

A generic definition of throughput is an amount of something per unit time. Please write down what is your VHDL implementation's throughput for (1) a Triple-DES encryption and (2) a Triple-DES decryption. Please also indicate the clock period of your design.

Just like software programming, there are multiple ways to implement a hardware function. Please describe one other way (different from your own implementation) to implement Triple DES's encryption and decryption algorithm that may or may not result in the same throughput. Explain how the other implementation would differ from yours in terms of resource utilization, power usage, and throughput.

V. Required Documents to turn in

You do not need to write a formal report, but you need to have the following contents in your submission.

1. Submit all the code (C code and VHDL files) with in-line comments pointing out the modifications you did in the files.
2. Brief explanation of the modification that you did to the main function and to the VHDL testbench.
3. Simulation results compiled into the .txt files described in the lab showing all keys, plaintext and ciphertext values.
4. Answers to questions in Section III (either in a README or a separate PDF).
5. Screenshot with waveforms verifying one encryption testcase and one decryption testcase. Please make sure the values shown on the waveform is legible.
6. Screenshot of the Triple DES resource utilization in Quartus.

Electronic Submission of Lab 3 Files

1. Place all files into a single folder and submit your work on Canvas.