DUE DATE: October 27th @ 11:59pm

DAV Lab 1

Setup & Introduction to Quartus

Introduction

Welcome to DAV! In this first lab, we'll be setting you up with all of the software that you'll need throughout this year. Mainly, we'll be using Intel's Quartus Prime Lite IDE to design, write, and program our FPGA projects. Additionally, we'll be using Questa to simulate our designs before we program them onto our FPGAs.

This document might look a little intimidating, but don't worry! There's a lot of verbose explanations and images to make sure that you have everything you need to get started easily, so it should go quickly.

Once you have everything set up, we'll walk you through creating and simulating your first project: decoding a secret message from your leads!

Reference Materials

Lab 1 Code   Verilog Syntax Cheat Sheet   EDAPlayground Guide

Getting Help

You can contact us either on Discord, through email, or in-person during our lab hours.

Tim Jacques:	TJ178#5214
Siddhant Gupta:	Condolences#1271

Our lab hours can be found at https://ieeebruins.com/lab

Checkoff Requirements

For this lab, you need to show us the final decoded message on your computer running Questa.

DUE DATE: October 27th @ 11:59pm

Part 1: Software Setup

To install the software you'll need this year, follow the instructions in these slides. In addition to installing the software, they'll also guide you through setting up your first Quartus project and simulation.

Windows:
Windows Quartus Installation

macOS (Intel):
macOS Quartus Installation

macOS (Apple Silicon): Unfortunately, Quartus Prime is a Windows-only x86-64 application. Therefore, the program cannot run on your computer (at least, not reliably), so you'll have to find another computer to use. You can write RTL in other environments such as edaplayground.com or VSCode and use extensions to simulate them. You should partner with someone else who can actually synthesize the code onto the FPGA. For this lab, though, feel free to use edaplayground.com. A usage guide is linked under Reference Materials.

Part 2: An Encrypted Message

In this portion of the lab, you will be introduced to using Questa as a simulation tool while applying combinational logic from Lecture 1. You will set up a Quartus project, upload the files we provide, implement the specified task, and then verify your correctness using Questa.

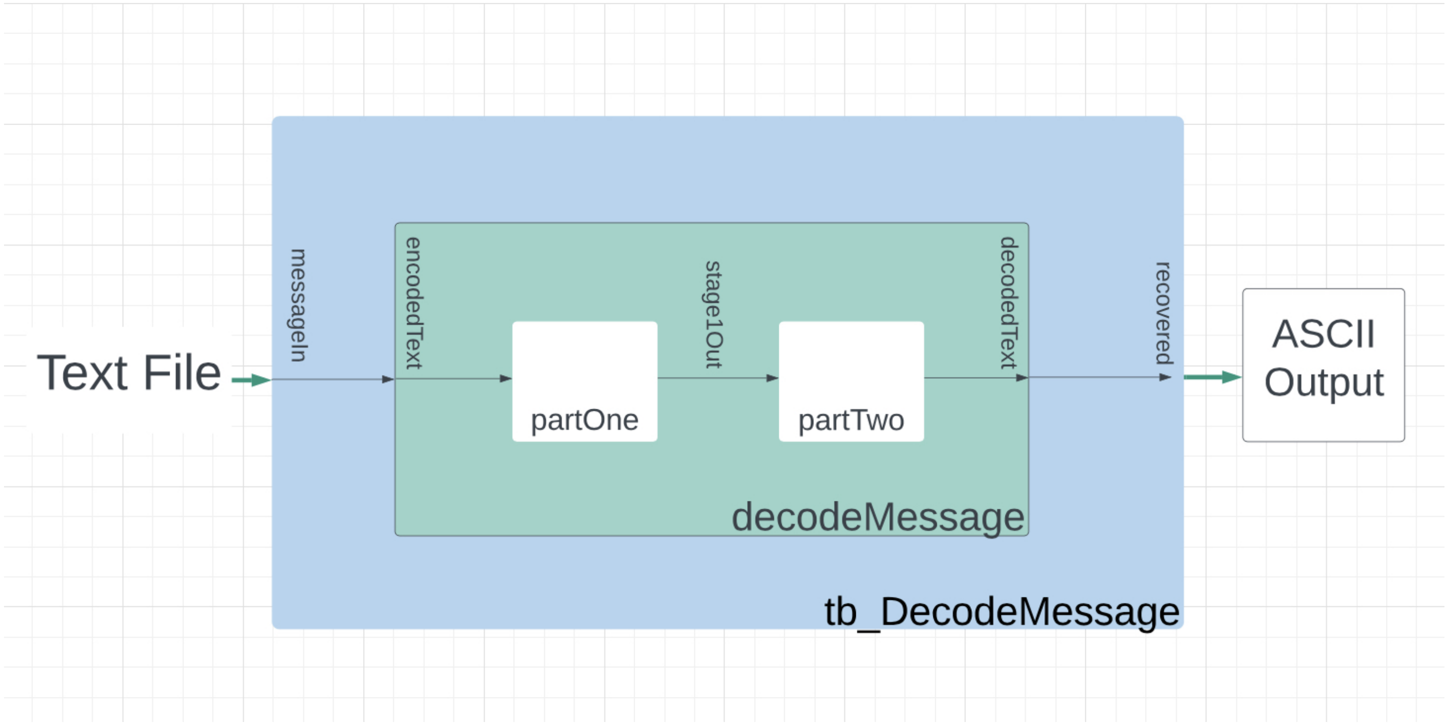
Suppose you want to pass on a message to a specific person while keeping it confidential for everyone else. One way to do this is to encrypt the message by doing various functions on it and using a key (which is a smaller string of data) to get it into a format that isn't really readable. Then, if the person you want to send it to knows what steps to do and also has the key you used, they can recover your original message!

Recall from lecture that all data on a computer is stored in binary. Text that we can read is typically interpreted from binary using an ASCII format. In this lab, you will be given some encrypted data in the form of a binary file with the objective of recovering the original text.

You can download files from the folder linked in Reference Material. The three components are the **encrypted_text.txt** file, **tb_DecodeMessage.sv** testbench file, **decodeMessage.sv** SystemVerilog file.

DUE DATE: October 27th @ 11:59pm

Follow the instructions below to complete this portion of the lab. You may find this diagram helpful in understanding the organization of the modules for this lab.



Step 1: Getting up and Running

Set up a new project (like you did in part 1) and then add these three files to the directory for your project. Then under Projects >> Add/Remove Files in Project you should add `tb_decodeMessage.sv` and `decodeMessage.sv` to your project. No need to add the encoded_text.txt to your project!

Before you start though, make sure that you change the filepath in `tb_decodeMessage.sv` to the location of `encoded_text.txt` on your computer, with forward slashes rather than backslashes. Quartus is a bit picky.

Next, under Project Navigator >> Files, you should set the testbench file to the top level.
Aside on the hierarchy: In Quartus, there should always be a top level design file that contains the rest of the modules in it. We will discuss this more in depth for Lab 3 but it is important to know for now that **if you want to run something in simulation, your testbench file must be the top level.**

Step 2: Hello World (but wrong)

You will learn how to make your own testbenches in lab 2. For now though, take a look at the testbench file and read through the comments that we have left. Note that our testbench is what reads from our message file and can display the data as ASCII characters. This is simply just a fun way to test our actual verilog design, which is the **decodeMessage.sv** file. We essentially feed in data from the file into this decode module (which currently doesn't compile!) and then we convert the output into text. Since we haven't decoded this message yet, it should just look like garbage for now.

Now run analysis and synthesis and run simulation through Questa. Look for the text output of the simulation

Step 3: Verilog!

Now, it is time to implement the logic to decode the text in the **decodeMessage.sv** file. For this, you should follow the instructions in the skeleton and feel free to refer to Lecture 1 or the syntax sheet for guidance.

You should periodically save (there is no autosave!) and run analysis and synthesis to make sure your code still compiles and that you catch errors early on. When you think you are finished, you can try running a simulation like in Step 2 again.

Step 4: Hello World

If your code is correct, the decoded message from the testbench should be very human readable (well, we aren't always the best at spelling…). Your very first line of output should be "Hello World" but the rest of them should also be English.

If this is true, congratulations! In this lab you learned how to install and set up a fairly complicated industry-standard development tool, navigate the confusing and cruel world of Intel, create projects in Quartus, write combinational logic in Verilog, and test it in simulation.

Now, you should screenshot your output from the simulation as well. Please fill out the google form to be checked off for this lab.