ECE4063 Large-Scale Digital Design

Project Description

S1-2025

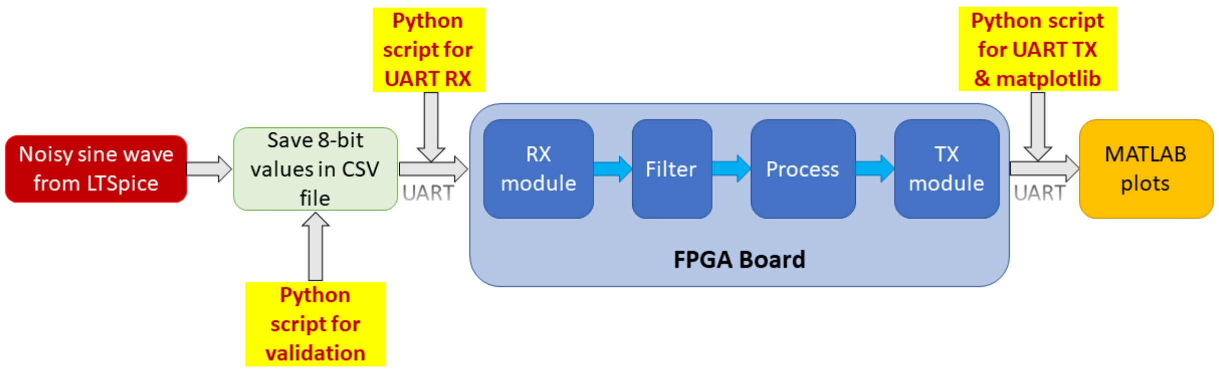
This project explores the integration of signal simulation, digital signal processing, and hardware-software communication through the development of a complete signal processing pipeline. Using LTSpice, students simulate an analog signal and model the behaviour of an analog-to-digital converter (ADC), generating realistic sampled data. This data is then transmitted over a UART interface to an FPGA, where a SystemVerilog-implemented FIR filter processes the signal in real-time. A Python script serves as both the control interface and a visualization tool, sending data to the FPGA and receiving the filtered output for analysis. Through this hands-on project, you will gain experience in mixed-signal simulation, HDL design and synthesis, FPGA interfacing, and data visualization.

Before you proceed, take note that this project is intended to be as open as possible. Hence, there may be more than one way to complete this project. However, if you used the assistance of any AI or genAI or anything similar, you must declare where, when, and how you used in your report.

Project Goals:

- Simulates analog signals using **LTSpice**
- Convert analog values to digital values **(ADC)**
- Transmits the data over **UART** to an FPGA
- Applies filter and **processing** on the FPGA
- Returns the processed data for visualization in **Python**

Overview and flowchart of the project


Python script for UART RX
Python script for UART TX & matplotlib
Noisy sine wave from LTSpice
Save 8-bit values in CSV file
UART
RX module
Filter
Process
TX module
UART
MATLAB plots
FPGA Board
Python script for validation

Part 1: Simulate Analog Signals Using LTSpice [5 marks]

In this part, we use LTSpice to generate a noisy sine wave. To do so, first open a new schematic in LTSpice. Place a sinusoidal voltage source (5V amplitude). The frequency of this source depends on how well it fits within the constraints of your digital system such as your sampling rate and the bandwidth of your filter on the FPGA. Do a quick research on this!

Place another sinusoidal voltage source to create noise with an amplitude between 0.20V to 0.50V (a randomly generate value will be provided to you). The frequency of this noise must be at least 10 times of your main sinusoidal source, to a maximum 15 times (again, a random value generated for you). Add a phase shift to further randomize your noise (random phase between 30° to 75°).

Quantize the voltage readings in your simulation to an 8-bit value (maximum 255). Research! Run the simulation and export the result as a CSV file with time and 8-bit voltage values.

Part 2: Convert analog values to digital values (ADC) [7 marks]

Because the FPGA will not be able to read analog devices, we need to ensure that the values in the CSV file has been properly digitalized. To do this, we validate the values in the CSV file using a Python script to ensure that the minimum value is 0 (representing -5V) and maximum is 255 (representing +5V). Any noise that cause your voltage to exceed ±5V will be capped to the ±5V range. Write a Python script to validate and ensure that the values in the CSV file is ready for UART transmission. Do a quick research on UART!

Part 3: Transmits the data over UART to an FPGA [8 marks]

Once we have the simulated values in digital format, we are ready to transmit the values to the FPGA. The method of this transmission is called UART. Firstly, write a SystemVerilog module to implement the UART RX logic. Write a new Python script to open the serial port communication and to send the CSV data to your FPGA board. Connect the FPGA to your PC/Laptop using the USB cable provided along with the DE10 board. You may need to check out the DE10 guide to learn how to connect the cable. We also strongly recommend you to use LEDs on the FPGA board to indicate that transmission has been successful.

Part 4: Applies filter and processing on the FPGA [40 marks]

In this part of the project, we take the 8-bit samples received over UART and apply a filter on the FPGA to reduce noise and to smooth the waveform. The filtered output will be sampled for further processing and then transmitted back to the PC for visualization. There are two types of filter that we recommend: (i) Finite Impulse Response (FIR) filter and (ii) Moving Average (MA) Filter.

After applying a filter to reduce noise from the signal, we will now process the samples to generate four different types of output signals: (i) Sine wave, (ii) Square wave, (iii) Triangle wave, and (iv) Frequency Modulation (FM). Use four switches and one-hot method to select the type of waveform. Determine your own selection priority if more than one is selected/switched on. Use this method to select the filter as well.

The sine is the filtered signal, which does not require further processing (remember that you simulated a sine wave in LTspice at the start of this project). The square wave and triangle wave requires a simple processing to produce the respective waves based on the same frequency as the simulated sine wave from the LTSpice.

On the hand, the FM wave will require the most processing. The carrier wave is a sine wave of fixed 10kHz, and the frequency of this carrier wave is modulated based on the filtered signal. A positive amplitude increases the frequency of the carrier wave, while a negative amplitude reduces the frequency of the carrier wave. The amplitude is the amount of change of frequency, e.g. +5V increase frequency of carrier wave by 5kHz, while -2V decreases frequency by 2kHz. The amplitude of the of the carrier wave must no change (±5V).

To complete Part 4 of this project, you will need to write a SystemVerilog design to perform (i) filter selection and apply the filter on the noisy signal that was received via UART and (ii) perform signal processing to generate a new output wave to be transmitted via UART. As a good practice and for better management of your project, you should separate the SV files of Part 3 and Part 4, as well as separate each major component in Part 4 into different modules.

Part 5: Returns the processed data for visualization in Python [10 marks]

The filtered and processed signal is transmitted back to the host PC over UART in this last part of the project. Python script is required to read this data and to visualize the results using matplotlib. Here's a simple guide for you to complete this part of the project: First implement UART TX logic on the FPGA to send processed signals (one of the four from part 4). Expand the previous Python script from Part 3 to receive bytes and store them (keep a copy of the previous Python script for marking purposes). Use matplotlib to plot both the LTSpice simulated signal and FPGA produced signal. Use this plot to compare the waveforms and determine if the filtering and processing has been successful.

Project Report [30 marks]

For this project report, you need to report only the tasks that you have completed for this project. If there are more than one group member that performed the same task, state the individual contribution of each member in the form of percentages, assigning a total of 100% for each task. Justification of the individual percentages in the shared task should also be explicitly elaborated.

Therefore, you must submit an individual report. The marks that you get from your project report may differ between each group member, however the marks that you get from the demonstration of the 5 parts will be the same for all group members. Contents of the shared task may be as similar as possible, according to the ratio. E.g. the person contributing 70% of the shared task should have 7/3 times more content than the person that contributed 30% to the shard task.

There is no fixed format for the report, but your marks are based on how well and how detailed you explain the tasks that you have contributed for this project. *The amount of contribution percentage does not directly reflect on how much you will score for your report*. However, ensure that your report give a good impression and be presented in a formal and professional manner.

Project breakdown

Group demonstration:	70 marks
Individual report:	30 marks
Total project marks:	**100 marks**