FPGA Guitar/Bass Effects Processor

Project Specification

Alex Florescu
Department of Computer Science
University of Warwick

1 Problem Statement

Current Problem

One of the main approaches to learning how to play an instrument is reproducing several pieces of music, preferably of increasing difficulty. In an attempt to recreate a specific sound, audio effects provide a noticeable difference and bring players a lot closer to making their instrument sound just like their favourite artist's. The guitar is the second most popular instrument in the world, with an estimated 700 million players, according to statistics[1] gathered by the guitar manufacturing company *Fender*.

The common procedure for adding audio effects to a guitar is either by purchasing *effect pedals* (hardware devices connected in between the guitar and the amplifier) or by purchasing software, which can be used by connecting the guitar to a computer through a separate audio interface.

Neither of these options is ideal. On one side, effect pedals offer the portability of a very small piece in a player's music setup but are often expensive. Additionally, the mixture of effects a player can create by using a couple of pedals is significantly limited. On the other side, using software to alter the sound of an instrument allows for a large range of adjustments (by simulating a pipeline of pedals or even replicating the signature sounds of popular amplifiers) and has the benefit of being upgradeable. While it might be a suitable decision for a music studio, a software-based setup becomes very inconvenient in a situation that demands mobility.

Solution

The aim of this project is to offer an alternative medium that combines the advantages of both sides by recreating a few commonly used effects on an **FPGA board**.

Xilinx, the biggest worldwide manufacturer of such devices, defines Field Programmable Gate Arrays (FPGAs) as *"semiconductor devices that are based around a matrix of configurable logic blocks (CLBs) connected via programmable interconnects. FPGAs can be reprogrammed to desired application or functionality requirements after manufacturing. This feature distinguishes FPGAs from Application Specific Integrated Circuits (ASICs), which are custom manufactured for specific design tasks. Although one-time programmable (OTP) FPGAs are available, the dominant types are SRAM based which can be reprogrammed as the design evolves."*[2]

The device that will be used for this project is a *Xilinx Zybo Z7-20*, a piece of hardware of small dimensions (8.8 x 12.2cm) with a maximum clock frequency of over 500 MHz. This will make possible the development of a real-time audio processing system, which will include a series of audio effects such as **delay**, **distortion**, and **reverb**. Implementing such a system on an FPGA will provide the advantages of an upgradable and highly customisable audio effects processor while maintaining the convenience of a compact device.

Building on the *ES2E3 Digital Systems Design* module content, in which an FPGA board was used to create a simple video-game, the **FPGA Guitar/Bass Effects Processor** project should constitute a challenging goal and a demanding learning experience.

2 Objectives

Mandatory Objectives

The core objective of the project is to design a series of mono audio effects (which are commonly used by guitar players) and implement them in SystemVerilog to produce a functional FPGA audio processor. This would consist of:

- Bypass option: permit the sound to pass through the system with no alterations.
- Delay effect: create an effect where a limited number of delayed copies of the sound are added on top of the input sound.
- Distortion effect: create an effect where the input sound signal is amplified and clipped, producing a *fuzzy* tone.
- Effect selector: implement a method to select one (or none) of the effects offered by the audio processor.

Optional Objectives

Depending on the progress and pace of the project, other effects and features can be added to the audio processor, such as:

- Reverb effect: adding multiple slightly delayed and lowered copies of the sound on top of the input sound.
- Effects pipelining: allow multiple effects to be selected at the same time, in a desired order.

Potential Objectives

If the prior objectives are accomplished far before the project deadline, some other features can be added to the audio processor, such as:

- Noise cancellation effect: removing an unwanted sound of a certain frequency from the input sound.
- Tuner: LEDs on the board can be programmed to provide information useful for tuning the guitar.

3 Methodology

Development methodology

Since the project is delimited by fixed deadlines, its duration can be estimated at 6 months. With the overall objective of the project being straight-forward and the tasks being rather independent, a rough plan can be created. However, the project requires a substantial amount of research before final decisions can be made about the development of the system. In other words, tasks can be arranged in an approximate timetable, but the priorities are subject to change over the first few weeks of the project.

To allow such changes, a mix between a plan-driven and agile approach is suitable, ensuring a balance between adaptability and steadiness. The project will be organised in 2-week sprints, with the intention that a new feature will be added to the audio processor at the end of each sprint. For each audio effect, filter, or feature of the system, we can outline a development cycle consisting of:

1. Research
2. Initial implementation
3. Reflection on potential changes
4. Final implementation
5. Testing

Code and documentation management

The local storage area where the project is developed will be regularly synchronised with an online GitHub repository. This must be done to prevent unwanted data loss and to guarantee a controlled coding environment that can be easily accessed by the project supervisor. GitHub will also be used to synchronise the documentation files with Overleaf[3] (a practical online tool for accelerated LaTeX compilation).

Testing

Since the project is developed in SystemVerilog, it will be thoroughly examined using testbenches. Each feature of the system will most likely be organised as an independent module of the audio processor. Therefore, the overall functionality of each component can be separately tested at the end of each sprint. The investigations should ensure not only a correct relationship between input and output signals but also adequate FPGA-specific statistics, such as *maximum frequency* and *latency*. After the project is considered complete, final testbench statistics will be collected to create a formal testing report.

4 Timetable

As mentioned above, the project development approach will consist of a mix of plan-driven and agile characteristics. A rudimentary Gantt chart of the tasks has been created, but it is expected to change in the near future as the project begins to take shape.

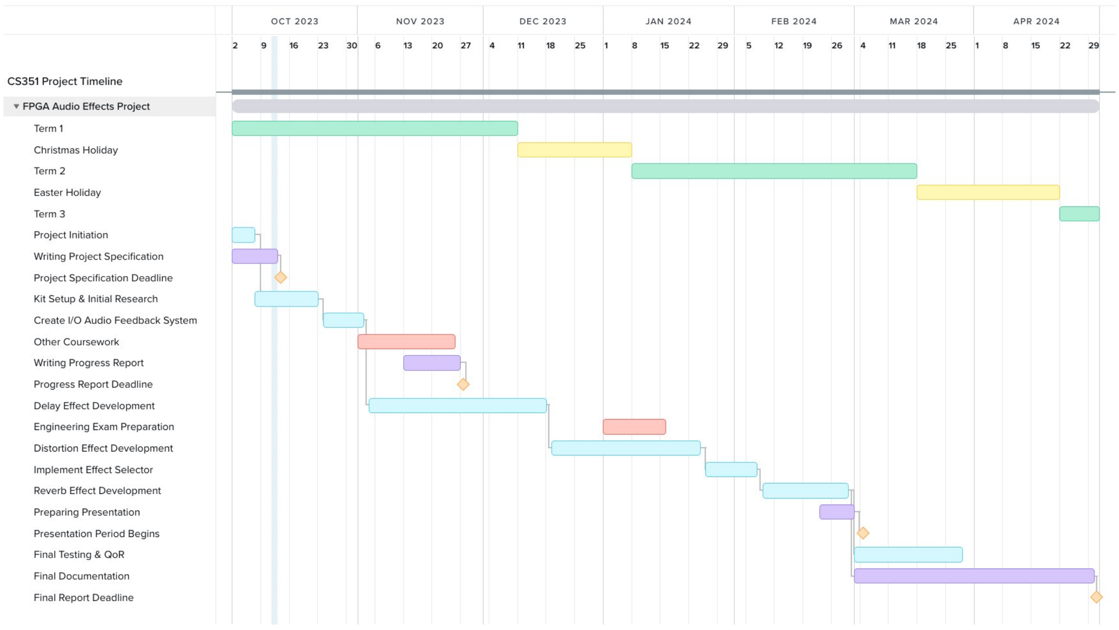

Figure 1: Project tasks organised in a Gantt chart

The timetable includes events that might interfere with the evolution of the system, and tasks surrounding these events have been assigned longer durations.

5 Risks and Resources

Potential Risks

In order to prevent unexpected setbacks in the development of the product, a set of possible risks has been considered:

- The PC used for developing the audio system breaks.
  - Use a cloud storage method for both code base and documentation, such as GitHub, to prevent any data loss.
  - Do not leave the PC unsupervised in public areas.
- The FPGA kit breaks, and development is stalled.
  - Avoid unnecessary transportation of the kit.
  - Only use the kit in a trusted and safe environment.
  - Use simulation testbenches as the main method of verification. Only perform hardware testing after simulations are denoted as successful.
- Student is ill or unable to work.
  - Respect a healthy schedule and maintain an adequate balance of rest and time dedicated to work.
  - Consider ergonomic recommendations, such as correct posture or appropriate room lighting.
  - Divide project objectives into mandatory and optional categories to allow deprioritisation of goals in the case of an unfortunate event.

Resources

- The system will be created in the SystemVerilog language. Code will be developed using Visual Studio Code and Xilinx Vivado. The latter will also be used for simulations and testbench creation.
- The developed code and documentation will be synchronised with an online GitHub repository.
- The FPGA board that will be used is a *Xilinx Zybo Z7-20*.
- The music equipment used is:
  - Harley Benton JB-20 Bass Guitar
  - Fender Rumble 40 Amplifier
  - 6.35mm Jack cables and 6.35mm Jack to 3.5mm Jack adapters

6 Legal, Social, Ethical, and Professional Issues and Considerations

Testing will be performed in a private environment, and the presentation will only be available to the Department of Computer Science and other staff members within the University of Warwick. The SystemVerilog system will not be published, and no data will be collected for the development of this project. Hence, there are no legal, social, ethical, or professional issues to be considered.

References

[1] Danielle Kittleberger. How many guitar players are there worldwide? (2019-2023 data), 2023.

[2] AMD Xilinx. What is an FPGA?, 2023.

[3] Overleaf. Overleaf - About us, 2023.