Midterm Research Project

Robert Ciliberto
ciliberto.r@northeastern.edu

Submit Date: 10/29/2020
Due Date: 10/29/2020

Abstract

This project is a culmination of the semester so far. It brings together the lessons of Object-Oriented programming in C++ and interfacing with the hardware of the DE1-SoC. By implementing two programs, a Music Player and a game of Snakes and Ladders, it reinforces what has been taught and solidifies that knowledge.

Introduction

The goal of this midterm research project is twofold. First, to utilize the hardware of the DE1-SoC to control the output of a speaker, and second, to utilize all that has been taught about C++ and object-oriented programming to design a game of Snakes and Ladders. This report will describe in detail the process and struggles that went into completing these tasks.

Video:

The video accompanying this report can be found here.

Software and Hardware Used

Hardware

- DE1-SoC

Software

- IntelliJ CLion (IDE)
- C++

Lab Steps

Tone Generators Using a Speaker

1. Create a text-based interface using a C++ terminal.
2. Have the user tell the program what they want to play
   (a) E.g. tell program to play C5 tone for a duration and then E5 afterwards.
3. Make sure the program can generate tones from C4 to C6.

Snakes and Ladders Game

1. Implement a game of Snakes and Ladders.
2. Be creative with the implementation
   (a) Decide which creative things to implement.
3. Make the board be at least 50 spaces long.

Lab Discussion / Results

DE1-SoC Library

To begin, it is necessary to implement the way to interact with the DE1-SoC. In previous labs, a `DE1-SoCfpga` class was implemented. It has been reimplemented in this lab. Additionally classes from previous labs that helped interact with with the devices on the board were reimplemented as well as new ones. The new ones allow for features that were not implemented in their own class or were not needed before. The `MPCoreTimer` class is a new class that allows one to control the MPCore timer on the DE1-SoC. The `ExpansionPort` and `Speaker` classes allow one to directly manipulate the JP ports via their pins and connect a speaker to the pins on the specified port. In this way, control of the DE1-SoC can be abstracted. With this bit of code written, it made sense to package it all up into a static library to be used in the main program. Perhaps it can be used in the future should the desire to create a new program on the board arise.

Tone Generators Using a Speaker

A key feature of the `DE1-SoCfpga` library is the ability to play tones on a speaker. The Speaker class allows for this implementation. Specifically, after connecting a speaker to one of the expansion ports and creating an instance of the Speaker class, one can call the `GenerateTone(...)` method with a tone that the user would like to play and the length of time they would like to play it for. With this abstracted out, the interface of the music player program can be implemented fairly simply. It allows for the user to type a series of instructions for the speaker to play written in a specific format unique to this implementation. This is implemented in the `MusicPlayer` class. The following is an example song that can be played via a command or pasting the script:

```
A5.18,D5.18,AsBf4.18,G4.18,G5.18,D5.18,AsBf4.18,G4.18,
    FsGf5.18,D5.18,AsBf4.18,G4.18,G5.18,D5.18,AsBf4.18,G4
    .18,G5.18,C5.18,A4.18,F4.18,F5.18,C5.18,A4.18,F4.18,E5
    .18,C5.18,A4.18,F4.18,F5.18,C5.18,A4.18,F4.18,F5.18,
    AsBf4.18,G4.18,E4.18,E5.18,AsBf4.18,G4.18,E4.18,DsEf5
    .18,AsBf4.18,G4.18,E4.18,E5.18,AsBf4.18,G4.18,E4.18,E5
    .18,A4.18,F4.18,D4.18,D5.18,A4.18,F4.18,D4.18,CsDf5
    .18,A4.18,F4.18,D4.18,D5.18,A4.18,F4.18,D4.18,A5.18,D5
    .18,AsBf4.18,G4.18,G5.18,D5.18,AsBf4.18,G4.18,FsGf5
    .18,D5.18,AsBf4.18,G4.18,G5.18,D5.18,AsBf4.18,G4.18,
    AsBf5.18,DsEf5.18,C5.18,FsGf4.18,A5.18,DsEf5.18,C5.18,
    FsGf4.18,GsAf5.18,DsEf5.18,C5.18,FsGf4.18,A5.18,DsEf5
    .18,C5.18,FsGf4.18,C6.18,D5.18,AsBf4.18,G4.18,AsBf5
    .18,D5.18,AsBf4.18,G4.18,A5.18,D5.18,AsBf4.18,G4.18,
    AsBf5.18,D5.18,AsBf4.18,G4.18,A5.18,AsBf4.18,G4.18,E4
```

```
.18,G5.18,AsBf4.18,G4.18,E4.18,F5.18,AsBf4.18,G4.18,E4
.18,E5.18,AsBf4.18,G4.18,E4.18
```

Bonus points if you know the song!

Snakes and Ladders Game

In this lab, a few creative features were added to the game. First, there are three print-out boards that can be selected, a 30, 50, and 100 space board. There can be as many computer players as the user likes. Each player takes a turn and when it is the user's turn they may press any of the four buttons on the DE1-SoC to make their move. The game features sound effects thanks to the previously implemented MusicPlayer class. These sounds include chimes for when the user rolls the dice, when the user goes up a ladder, when the user goes down a snake, when the user wins, and when the computer wins.

Analysis

Tone Generators Using a Speaker

There were a few challenges when trying to implement the MusicPlayer class. Specifically, language specific details such as parsing the user's input was not as easy as it is in other languages like Java or Python. While there is room to improve, such as checking input edge cases, the menu is navigable and the user can play music as they like. Additionally getting the speaker to create the right tone required a bit of math to figure out how many cycles at the specified frequency would fit into the desired time. Some time was also spent creating the song in the previous section to demonstrate that theoretically, a full song could be played via this method.

Snakes and Ladders Game

Coming from a Java background, C++ proved to be much more difficult to implment true object-oriented design. However, after some online research (from the help of StackOverflow) it was fairly simple to transfer design ideas between the languages' frame of minds. The Snakes and Ladders game was implemented with a Model-View-Controller framework. Although the current implementation does not utilize VGA graphics, it is certainly possible, and made easy thanks to this framework. Special thanks to `www.sparklebox.co.uk` for providing free samples of Snakes and Ladders boards on which the boards that are playable in this game are based. Attached is a PDF they supply freely that has three Snakes and Ladders game boards.

Implementation wise, a strong emphasis on abstraction and organization was made. That is why the code for the DE1-SoC was abstracted out into its own sttaic library. Additionally the code is organized so that it is easy to understand individual components such as the Board, Player, and Snakes and Ladders.

Conclusion

This project was a culmination of the semester so far. It brought together the lessons of Object-Oriented programming in C++ and interfacing with the hardware of the DE1-SoC. By implementing these two programs, a Music Player and a game of Snakes and Ladders, it reinforces what has been taught.