Project plan: Piano Hero

25.09.2018

Group 15: Aiden Tesolin, Tom Grooters, Jakub Myśliwiec, Tamerlan Askarzade , Andrew Dziouba, Varun Sudhakar

[1] Image: https://pixabay.com/en/piano-keyboard-hands-music-play-2706562/

Introduction **3**

Description of Hardware and Software **3**

DE1 SoC - Hardware 3

Raspberry Pi - Hardware 4

Linux (Raspbian) - Software 4

VHDL - Software 4

MiDi Keyboard - I/O (Hardware) 5

Screen - I/O (Hardware) 5

Application **5**

Description 5

Specification 6

Design (mapping onto hardware and software) **6**

Implementation **7**

Testing **7**

Required Tooling **8**

Required Hardware **8**

Planning **8**

Completing the Project Plan 8

Conclusion **9**

Introduction

In the current times technology is moving forward at an increasingly rapid pace. More and more products are developed and sold, which leads to high competitiveness. In order to stay relevant in such an environment the products require close cooperation of software and specialized hardware. This cooperation is what this project will focus on and the goal is to have a Raspberry Pi running in cooperation with the DE1 SoC FPGA in order to accomplish a certain task. A user interface will need to be designed. The DE1 SoC will do some of the more resource intensive tasks and sensors will be utilized as input for the application. With this hardware it should also be possible to achieve real time behaviour.

The best projects start with a well devised plan for all the functionality. The requirements are known in advance and the planning is realistic. Additionally it is vital to keep the available resources. In the case of this undertaking, 6 students with varying background will essentially need to make this application within 2 weeks with some previous preparation, work together and communicate effectively in an interdisciplinary environment. The following part is an attempt at such a plan for the described project.

Description of Hardware and Software

DE1 SoC - Hardware

The DE1-SoC (System on a Chip) is an FPGA (Field Programmable Gate Array) which is basically a bunch of Transistors which can be arranged and programmed in any way. Resulting in a hardware executable program. The FPGA is built out of 3 basic components: Logic, Programmable interconnects and I/O. As with any simple processing system. The I/O handles inputs and outputs which are then relayed via the interconnects to the logic which processes it and sends back a result to the I/O. The Programmable interconnects are what define an FPGA since they route all the communication between the logic blocks and the I/O blocks. These interconnects are usually also the slowest part (bottleneck) of the FPGA. The Logic blocks consist mostly of a Lookup table (LUT) and a flip-flop. THe flip flop is used as data storage and the LUT is used as any 4 input logic system, 16-bit memory or 16-bit shift register. The I/O handles the inputs and outputs of the fpga, which are mostly configurable to the users needs.

FPGAs are usually used for designing prototype integrated circuits to find errors before the circuits are mass produced. As objective of this project, the FPGA will be used to offload as much as possible of the workload off of the raspberry, as a Support processor or an Application specific integrated Circuit (ASIC). More specific to the idea chosen the FPGA will function as a synthesizer or sound card. The FPGA will be tasked with generating sound from a basic input from the pi and processing it.

Raspberry Pi - Hardware

The Raspberry Pi is a small computer that is build onto a single board. Despite what the small size and low price would it seem, it is still a relatively powerful machine. Making it hugely popular in wide areas of interest including hobbyists and educational institutions.

In our applications the Raspberry Pi will handle the inputs of the Midi keyboard, the display and logic of the game while sending the notes/sound ques over to the DE1 SoC to be further processed and turned into the actual sound output.

Linux (Raspbian) - Software

The operating system running on the Raspberry Pi will be Raspbian. Which is a open source Debian based Linux distribution running under the same licenses. This OS is widely used on the Raspberry Pi since it comes with a collection of tools/programs that is pre selected and pre configured to work well with a Raspberry Pi. It comes in two versions; The lite package, which does not include a desktop environment, And the raspbian desktop package which comes with it's own desktop environment called PIXEL

VHDL - Software

The Language used for programming the DE1-SoC is VHDL (**V**HSIC (**V**ery **H**igh **S**peed **I**ntegrated **C**ircuit) **H**ardware **D**escription **L**anguage) which can be used for modelling and testbencing (in the same language) designs inside the program. It's highly used to describe digital and mixed-signals (analog and digital), for instance FPGAs and IC's. VHLD is a highly typed language that contains wide range of data types, the ones like numerical, logical, character, time and arrays. After periodical change in standardization the language evolved quite a bit since it's release in 1983.

A text model is written using VHDL that describes a logic circuit. After the model is processed by a synthesis program, a simulation program can be used to test the circuit and implement the design. Which is basically called testbench. Some of the key properties of VHDL is that supports hierarchy. Various design flows are supported (top-down, bottom-up, mixed). However, during this project top-down flows will be used. Moreover, both synchronous and asynchronous systems can be implemented. Nevertheless, asynchronous systems are most of the times ignored since it is too hard to get it to work and test it. FSMs (finite state machines), boolean equations and algorithms can also be realised using VHDL. There are plenty of advantages of this program that easen the life of its user, however, it is a strongly typed and everything should be stated explicitly. Besides that, VHDL handles higher level of abstraction, second sourcing is easy and LSI modelling is not hard. The program has no size restriction and one of the most useful properties that it is concurrent. Hence, processes are executed concurrently. Therefore, ordering is not important, and processes are internally executed sequentially.

MiDi Keyboard - I/O (Hardware)

A MiDi keyboard (**M**usic **I**nterface **D**igital **I**nterface) is widely used in the music industry to create different sounds and melodies. They are typically used along with a software that helps the MiDi keyboard synthesize sounds of different octaves, pitches and so on.

The MiDi keyboard itself produces no sound when switched on since it has no sound engine, hence when connected to a DAW (Digital Audio Workshop) such as a computer with softwares such as FL Studio or Ableton Studio.

The MiDi keyboard will be the user input for the Piano hero game. The MiDi keyboard will further be connected to the Raspberry Pi that will process the input.

Screen - I/O (Hardware)

An LCD display will be used during this project to output the games visuals. The screen that is used has the VGA resolution standard which is 640x480. Not much more is known about this screen, since it hasn't been received yet, but judging from the resolution, it's not going to be much bigger than a small laptop screen.

Application

Description

For the project we decided to take upon us developing of a music game. The piano hero.

This will be a game where a player will play a song on the piano, based on the notes displayed on the screen. The piano output will be cross referenced with what the game expects the player to play, the notes played by the player will be sent to an audio output from the DE1-SoC. The DE1-SoC creates the sounds for the song based on the input with an 8-bit retro style synthesizer. This is comparable to the game Guitar hero, while playing songs from the likes of, for example, retro mario games.

The Game will display a dynamic stream of notes on the display from a song, the player will have to input the right key within the right time period for the game to accept it, if not then the game will output a fail sound. If the sound is accepted then the appropriate sound ID will be outputted. Some visuals will be accompanied with the stream of notes on the screen to indicate if the key is accepted or not. All the sound output is sent to the DE1-SoC which runs a synthesizer to generate the sounds corresponding to the ID. This sound is created and output to a speaker. Optionally a Joystick will be added and passed through the DE1-SoC to navigate the Menus of the game.

Additionally there will be a free play option where no keys are expected and the use will be able to freely play the piano and hear a retro sound generated by the DE1 SoC board.

Specification

In terms of functional or quality requirements the game shall satisfy the following:

- Delay from the press of a key on the keyboard to hearing the sound and updating the GUI is no more than 25 ms
- 95% of people should be able to read the GUI and press the indicated key
- 90% of tested people should be able to play a simple recognizable tune on the piano after 5 minutes of learning
- There will be at least 3 songs that are available to play in the game

With enough time and luck it may also be possible to fulfill the following extended requirements:

- It is possible to import songs to be played from midi sequence files.

Design (mapping onto hardware and software)

Raspberry Pi - Will be to process the input data from the MIDI keyboard and output the data to the DE1-Soc which will play the respective notes. Also it will be used to process the game logic and GUI, which would be output to the VGA Display.

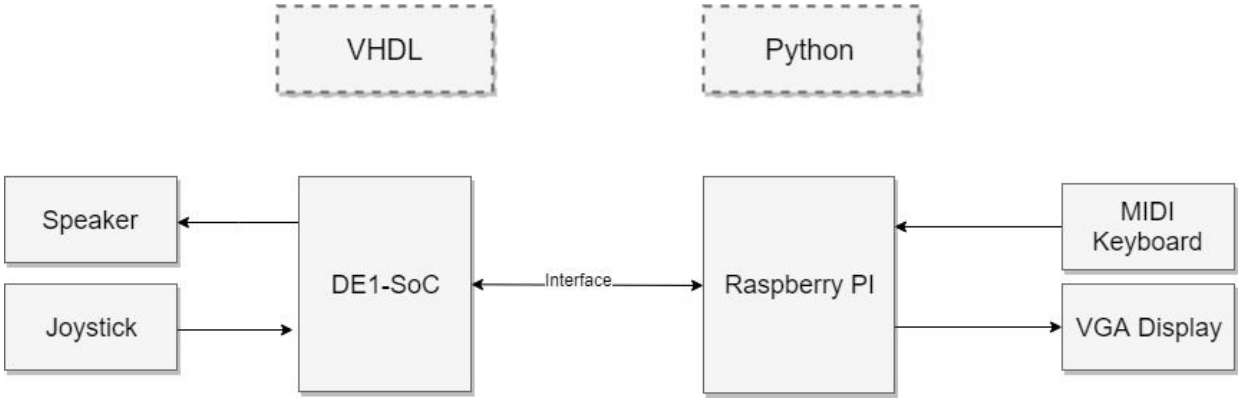
MIDI Keyboard - Its keys will be used as input for the game.

VGA Display - It will be used to display the output for the game.

DE1-Soc - Will be used to generate the notes for the game and would be used to connect to the speaker and Joystick as input and output.

Joystick - will be used as a input to navigate the games main menu, which could also be done on the MIDI Keyboard making the joystick optional.

Speaker - would be used to output the sounds generated on the DE1-Soc.



Implementation

The Raspberry PI will take the input from the MIDI keyboard and process the data to be used as part of the game logic, and sent to the to the DE1 SoC to be processed and turned into sound output. The game logic, GUI and data processing on the Raspberry PI will be implemented using python.

The DE1-SoC will take data from the Raspberry PI and generate corresponding notes using a synthesizer, which will be implemented using VHDL software.

Testing

Multiple parts of the system have to be tested separately and verified if they work according to design.
Some test include: Interfacing between Raspberry Pi and Midi keyboard, Interfacing between the Raspberry Pi and the DE1-SoC, inputs from the Joystick, output to the speaker, Output to the Display and Application testing.
The Application will be tested by playing and trying to “break it” to find bugs, this is the only way of testing that part of the system.
The interface with the MiDi Keyboard will need some simple code which reads out the inputs or maybe some software which reads the inputs, this can be used to verify if the inputs are functioning as expected.
The inputs of the Joystick will have to be tested in a similar way to the MiDi keyboard, but there’s probably no software which can already read it and it will have to done with a simple code in, for example python.
The output to the speaker will be tested by generating a sound on the FPGA and outputting to the speakers and see if there’s a sound.
The Output to the display will be tested by drawing something on the screen in software and checking whether this is the desired image on the screen.

There's currently no known way to test the interface between the DE1-SoC and the Raspberry Pi, this will be the first priority to find while trying to establish a connection.

Required Tooling

The following software will be necessary to complete and build the developed application

- Python compiler
- Appropriate python GUI framework
- Driver for the MiDi keyboard
- A display driver compatible with the Raspberry Pi
- VHDL Compiler
- An IDE for VHDL to work in
- Software to map the code onto the DE1-SoC (Synthesis)

Planning

Completing the Project Plan

Varun - Raspberry Pi connection to DE1 SoC and MiDi decoding

Tom - Raspberry Pi connection to DE1 SoC and MiDi decoding

Jakub - Game Logic(Checking input, showing which key needs to be pressed at what time, etc.)

Tamerlan - GPIO connection between the Raspberry Pi and the DE1 SoC and help with sound generation and processing

Andrew Dziouba - Sound generation and output using the DE1 SoC

Aiden - Designing and implementing the GUI for the game

The project management will still be done in a loose agile fashion and the task assignment is loose. Everyone could be doing anything at any point. Tasks not foreseen are likely to come up and this approach was designed to deal with these types of situations.

Conclusion

This will be a challenging project and it is going to involve plenty of learning. Both on the technical side as well as on the soft skills (communication and group work). Many people will need to learn a way of working with on the same software at the same time. Many differences in the specific language used to refer to various things will need to be clarified and other problems with interdisciplinary will be dealt with.

Additionally, plans are subject to change. Some aspect of the application may have been missed, a part of the implementation is taking too long, someone is not able to do the job previously seeming perfectly doable. All of the unforeseen problems will need to be dealt with within the resources available. In the worst case some planned functionality may not be implemented.

However, it will be all worth it since plenty of knowledge will be gained, experiences had and, hopefully, a great game providing entertainment will be built.