September 2014

Roel Plompen (1496808), Arnold Averink (1493841) & Willem Siers (1424661)

Faculty of Electrical engineering, mathematics and computer sience

Project Plan

# Inhoudsopgave

[Inhoudsopgave 1](#_Toc400462631)

[Preface 2](#_Toc400462632)

[Hardware Description 2](#_Toc400462633)

[Raspberry Pi Model B 2](#_Toc400462634)

[DE1 SoC 2](#_Toc400462635)

[Raspberry Pi VS DE1 2](#_Toc400462636)

[Software Description 3](#_Toc400462637)

[Rasbian (Debian Wheezy) 3](#_Toc400462638)

[Application 3](#_Toc400462639)

[Description 3](#_Toc400462640)

[Specification 3](#_Toc400462641)

[Design 4](#_Toc400462642)

[User-interface 4](#_Toc400462643)

[Input-processing 4](#_Toc400462644)

[Raspberry Pi to DE-1 SoC communication 4](#_Toc400462645)

[Sound Synthesis on the DE-1 SoC 4](#_Toc400462646)

[Audio processing 4](#_Toc400462647)

[Direct user input 5](#_Toc400462648)

[Audio Output 5](#_Toc400462649)

[Implementation 5](#_Toc400462650)

[User input and user interface 5](#_Toc400462651)

[Audio processing 5](#_Toc400462652)

[Recording the Audio 6](#_Toc400462653)

[Testing 6](#_Toc400462654)

[Testing of the DE-1 6](#_Toc400462655)

[Testing of the Raspberry Pi 6](#_Toc400462656)

[Testing of the entire solution 6](#_Toc400462657)

[Required Tooling 6](#_Toc400462658)

[Planning and definition of tasks 6](#_Toc400462659)

[References 7](#_Toc400462660)

# Preface

In the project plant an application involving a Raspberry Pi and DE-1 FPGA Development board. First a short description of the Raspberry Pi and DE-1 SoC are given. Following that a description of the application, as well as its requirements. following that, a design plan and implementation of this plan is given. last, a division of tasks and planning is provided.

# Hardware Description

## Raspberry Pi Model B

The Raspberry Pi is a small (85.60 mm × 56 mm) computer developed by the Raspberry Pi Foundation mainly for educational use. The Pi has a Broadcom BCM2835 SoC containing a ARM1176JZF-S 700 MHz processor and an embedded VideoCore IV GPU. Our model of the Raspberry Pi Model B has 512 MB RAM. The processor uses the ARMv6 instruction set [1].

## DE1 SoC

The DE1 is an development board based around a cyclone V FPGA [2]. The power of this board is in its FPGA, which enables it to perform certain tasks a lot faster than a regular processor would be able to do. For this project, the FPGA will work together with the Raspberry Pi, which will function mainly as an user interface. The Cyclone 5 SoC is split up in 2 parts. It contains an hardwired dual core A-9 processor, and an FPGA with 85K logic elements.

The ARM processor has 1 GB of DDR3 SDRAM. While the FPGA part of the chip has 64MB of SDRAM. The memory is usually used in order to buffer data, or store program data, not as a permanent storage solution.

The DE1 also is well equipped on the interface part. It has a lot of interfaces, such as Ethernet, USB, audio and video. But it is also able the handle protocols such as I2C and SPI.

## Raspberry Pi VS DE1

Even though the Pi and the DE1 have a lot of similarities, they are both good at very different things:

|  |  |  |
| --- | --- | --- |
|  | Cyclone V FPGA [2] | Raspberry Pi [1] |
| Clock speed | 50MHz | 800MHz |
| Multiply operation (clock cycles) | 1 | ? |
| Amount of parallel multiply operations | 174 | 1 |
| Communication speed (Ethernet) | 1000Mb | 100Mb |
| memory | 64MB | 512MB |
| User interfaces | UART, USB2.0, Ethernet, PS/2, IR Emitter/Receiver | USB, Ethernet, SSH, UART |
| Video | VGA | HDMI, Analog |
| Audio | Analog | Analog, HDMI |

As we can see, the Pi’s raw processing power is not that interesting. However, the Pi excels at its video output (HDMI vs VGA), and it’s the software support (libraries, Linux OS) make it a viable option to be used as a user interface. However, considering that the DE1 has a hard processor system which is better than the Pi, the DE-1 is capable of completely replacing the Pi.

# Software Description

## Rasbian (Debian Wheezy)

Raspbian is a free open source operating system developed by the community and optimized for the Raspberry Pi [3]. It is based on Debian Wheezy a very popular Linux distribution. Ubuntu, the most used Linux distribution, is also based on Debian.

Raspbian has many pre-installed packages so users can immediately start using their pi.

A few of the examples of the pre-installed programs are the GNU C/C++ Compiler, python and the java development kit. With these programs you can easily develop your own software for the Pi.

Many developers have developed specific software for the pi. This has resulted in free versions of normally commercial software like Minecraft Pi edition and Mathematica for the Raspberry Pi.

# Application

## Description

The application will be a multi-channel synthesizer, that is to accept input from external devices. This input will be used to instruct the application which sounds are to be synthesized. This input will be either in direct mode, such as an ordinary computer keyboard whose input will immediately be processed and played back, or the input will not be directly processed, but rather stored in main memory, where it can be edited and then, when appropriate, parts of the stored instructions will be used to synthesize sound. Optionally the application will implement some sort of post-processing on the outputted sound.

## Specification

The application:

* will run on the Raspberry Pi model B running Raspbian, in combination with the DE-1 SoC
* will accept input from a (midi-)keyboard over usb
* can play back up to 4 channels simultaneously
* allows selection of input mode from the user-interface
* allows indirect input by supplying input-data from a memory device
* will use the DE-1 SoC's Line-out to output 24-bits audio

# Design

The design of the application will have to take advantage of all the different components our total system will have, which are the Raspberry Pi, the DE-1 SoC, a way to control the application (also known as the interface), and the different forms of input and output.

Also with the design it is important to think about the different ways to communicate between those components, which can be a physical medium like interfacing between the DE-1 SoC and the Raspberry, but also the interface between the user-interface and the actual application, so that any input that the user gives to the application will actually be available to the application

## User-interface

For our user interface we decided on using a web-interface to keep the interface flexible and maintainable. This interface will allow the user to select several parameters that will be used by the application to process the input that it is given through the keyboard. These options will include:

* the channel on which the received input will be outputted
* which procedure will be used for synthesizing the sound for each channel (e.g. the waveform to produce and other parameters for sound creation)
* volume options per channel and a mute option
* per channel a post-processing effect to be applied
* select sequence data for the remaining channels that do not receive direct input from the keyboard.

## Input-processing

The direct input from the keyboard will be processed by the Raspberry Pi, because it is easy to interface with this device and external devices, and also because the application will have to generate signals to synthesize the audio based on both the input of the keyboard and the currently set parameters that are set by the interface.

## Raspberry Pi to DE-1 SoC communication

For the communication we will use the I2C interface to send a stream of data to be processed by the FPGA to the DE-1 SoC. This is a stream in the sense that there will be a lot of data being sent and set to the DE-1 SoC's registers. The DE-1 SoC will have it's registers set by the Raspberry Pi's program, and the DE-1 SoC will just read the set values from these registers and use it to create the sound. To simplify matters, there must be a single format in which the data will be set to the DE-1's registers, so that there does not have to be made a difference between reading data supplied by direct user input through a keyboard, and data that was specified by a sequence.

## Sound Synthesis on the DE-1 SoC

The actual sound synthesis will be the responsibility of the FPGA on the DE-1 SoC. This board will listen to the clock signal of the processed. This data will be set by the C program running on the Raspberry Pi, and this program will also serve the clock signal. This way the FPGA can be a 'dumb' device, which doesn't have to worry about opening and reading files or a data sequence, but only has to do one repeated task based on the values that were set to it's registers.

## Audio processing

The audio will be processed on the Altera FPGA. The FPGA performs the requested operations on generated sound signals. The Raspberry Pi “tells” the FPGA which sound waves should be generated on which channels.

After performing the effects on the different signals, the channels can be mixed together to one.

The available effects include the following:

* Echo
* Reverb
* Volume manipulation

After the effects are added to each channel, the channels can be mixed together to form the output of the entire system. The output is converted into sound which is transmitted on the line out of the FPGA.

## Direct user input

The input that will be used to directly play notes will be of a midi-keyboard, connected through usb. This input will be processed directly by the program on the Raspberry Pi, and it's data will be used to set-up the registers of the DE-1 SoC, according to the options that were set by the user-interface.

## Audio Output

For the audio output we will use the standard analog audio output on the DE-1 SoC. This output will be connected to an amplifier and a speaker to actually hear the sound. If wanted, this signal could be fed back into the Raspberry Pi for further processing or recording, however this is out of scope for this project.

# Implementation



This is an overview of all the forms of communication in the system. It will also be possible to download the file from the Raspberry Pi after saving it to the SD Card

## User input and user interface

To access the GUI of the application the user can use the Raspberry Pi directly or use a separate computer. To achieve this we will use a webserver using python so the user can use the application directly through the browser without installing any software. By using python we can easily load c library’s. This is required to get low level access to the system so we can communicate with the DE-1.

The user will be able to use the web interface to select effects and send midi files to the Raspberry Pi. These midi files will then be processed using the effects determined by the user. It is also possible to directly insert real-time music by using a USB-keyboard if a keyboard is connected to the Raspberry Pi. It will properly not be possible to use the keyboard of an external device. The midi file or the real time input will be converted to a protocol we will define.

## Audio processing

The Raspberry Pi Uses a I2C interface to request operations on the FPGA. After the FPGA has received a command, it will initialize the 4 channels it has available. For each of these channels, separate effects (if any are requested), are performed. After performing the effects on the appropriate channels, the different channels are combined into one, and outputted to the line out.

## Recording the Audio

Optionally we want to add the ability to record the audio instead of playing it. The user will then use the web interface to request the audio file while submitting a file and let the Raspberry Pi record the output of the FPGA. The Raspberry Pi will then record the data using a line in port of an external USB-Sound Card. The recorded file will be stored to the SD card of the Raspberry Pi. The user can then download the file directly using the web interface. The output files will be saved in a file format like mp3 of wav.

# Testing

## Testing of the DE-1

The application for the DE-1 is going to be debugged using simulation software, which simulates the contents of the VHDL hardware description. From that, the simulated results can be compared with what is to be expected from the application. This is done by writing separate test benches for the different functions.If the simulation and expectation match, the program can be synthesized to work on the DE-1 SoC.

## Testing of the Raspberry Pi

To test the software running on the Raspberry Pi, we will only create explicit unit tests for the HTTP requests that will occur between the webpage and the webserver to see if the communication works successful. For the rest it's a lot more efficient to just provide user-input by ourselves and inspect if the obtained behavior is what we wanted, than to write tests to automate or simulate this sort of input (think about the input from the midi-keyboard) and automatically observe whether the results are according to the specified expected results.

## Testing of the entire solution

Testing of the entire solution should be fairly straight forward. First some input is selected and an expected output from the entire application is determined. That input is than inserted into the application. From that point, the data is followed throughout the application, so the output of the Raspberry Pi/ Input of FPGA is checked, and the result returned from the FPGA, is checked against the expectations. If the expectation and measurements match, the application is built successfully. If not, we should be able to determine where along the line it went wrong, so we can improve that part of the chain.

# Required Tooling

The only tooling required for this project are the connections between the Raspberry Pi and the DE1. A power supply is mandatory. A display (computer monitor) will be used to display the GUI from the Raspberry Pi. Optionally, a LAN of WIFI connection in order to reach the web interface. Finally, some loudspeakers/headphones are required in order to listen to the result!

If we want to record the audio generated by the FPGA we will need an USB Audio Card, because the Raspberry Pi audio port cannot be used to record Audio signal. As a fall back method we can also use the USB port directly, although this will require that we implement the sending of the audio signal in two different ways.

# Planning and definition of tasks

|  |  |  |
| --- | --- | --- |
| Task | Student(s) | Deadline |
| Interface Raspberry and (MIDI) Keyboard | Willem | 28-10-2014 |
| (optional) Text base GUI | Arnold & Willem | 21-10-2014 |
| Web based GUI | Arnold | 5-11-2014 |
| I2C (Raspberry side) | Willem | 30-10-2014 |
| I2C (DE1 Side) | Roel | 21-10-2014 |
| USB (Raspberry side) | Arnold | 30-10-2014 |
| USB (DE1 side) | Roel | 31-10-2014 |
| FPGA Programming (effect algorithms) | Roel | 4-11-2014 |
| Storing results (Raspberry side) | Arnold | 4-11-2014 |
| Testing | everybody | 4-11-2014 |
| Making the report movie | everybody | 5-11-2014 |

# References

|  |  |
| --- | --- |
| [1] | E. 14, „Raspberry Pi Datasheet,” 13 January 2014. [Online]. Available: http://www.element14.com/community/servlet/JiveServlet/previewBody/65470-102-1-287848/Raspberry-Pi%20Technical%20Data%20Sheet.pdf. [Geopend 7 October 2014]. |
| [2] | Altera, „DE-1 SoC User Manual,” 14 March 2014. [Online]. Available: http://www.terasic.com.tw/cgi-bin/page/archive.pl?Language=English&CategoryNo=167&No=836&PartNo=4. [Geopend 7 October 2014]. |
| [3] | „Raspbian,” Raspberry Pi Foundation, [Online]. Available: www.raspbian.org. [Geopend 7 October 2014]. |