UNIVERSITÄT PADERBORN

Die Universität der Informationsgesellschaft

Faculty for Electrical Engineering, Computer Science and Mathematics
Department of Computer Science
Computer Engineering Group

PG Soundgates

Project Plan

Author:
Martin Boonk
Caius Cioran
Lukas Funke
Hendrik Hangmann
Andrey Pines
Thorbjörn Posewsky
Gunnar Wuellrich

Supervisor:
Prof. Dr. Marco Platzner
Jun.-Prof. Dr. Christian Plessl
Dipl.-Inf. Andreas Agne

July 13, 2013

Contents

1 Introduction

1.1 Introduction

— Von Hendrik, mit anderem Kram mergen —

"'Soundgates Interactive Music Synthesis on FPGAs"' is a project initiated by the Computer Engineering Group of the University of Paderborn, aiming to synthesise music on modern FPGAs. Furthermore, one or more users should use this interactive system, by means of manipulating the synthesised music with advanced sensors such as the acceleration sensor in modern smartphones.

1.2 About this document

This document is the project plan of the project group "Soundgates" at the University of Paderborn. It introduces the topic of our project group in Chapter 1 and the goals we want to achieve in Chapter **??**. Chapter **??** covers systems similar to what we are going to create, escpecially the software called MAX. Chapters **??** and **??** describe how the group will organize itself and the groups' milestones.

Not only should this document serve as a basis for the later evaluation and grading by our professor, but also as a reference for our group during the main working phase. This working phase runs from ??.2013 to ??.2014

1.3 Definitions

The following table displays some definitions, that will be used throughout this document.

Term	Definition
Composite Component	Definition
Component	A basic building block to generate music ??Haben wir uns hier auf Block als Bezeichnung geeinight? Component eher im Sinne von Softwarekomponente
Editor	The Editor is used to create a patch out of components to generate synthesizable code which can be put on a FPGA
FPGA	Field Programmable Gate Array
Patch	The entire system which consists of Components and Composite Components. A set of single Components can build a new Component
Port	The interface from one Component to another one
Simulation	The developed patch is played through the PC speakers

1.4 Project outline

- Creating an editor for synthesizers. - Components of the synthesizer implemented in hardware (and software for simulation purpose) - Implementation of generative music concepts ——- Warum wollen wir das eigentlich tun? Was ist die Motivation dahinter (zumal es das aus Oslo ja aschon in Software gibt)

1.5 Introduction to sound synthsis

- Artifical generation of sound - Generation of basic waveforms - More complex and rich patterns by methods like additive/subtractive/... synth - Further addition of filters etc

- Originated from analog synthesizers, nowadays mostly digital. Software for general purpose PCs exist

- Building patches: job of a sound designer, rather than a musician

1.6 Generative music

— Stichpunkte Gunnar: —

- Creatioin of music depending on user interaction - Users do not need to be musicians - Playful approach to making music - tightly connected to sound synthesis

1.6.1 Introduction

As pointed out in [**?**], there are many cultures where musicical experience is defined by people performing and people perceiving music. The only way to slightly exert influence on the performer's music is by cheering, shouting, etc. on a concert. The gap between performer and perceiver reaches its peak in the context of recorded music like CDs or MP3 files, where is no chance of manipulate the music. Actually, there is a rising trend for interacticing with music on your own or with a familiar social partner, like in the

video game "'Guitar Hero"' [?, ?]. Even without any musical knowledges or talent, it is more and more possible to interact, manipulate or even create music. Generative music combines the opportunities of making music without having knowledges of how to play an instrument and explicitly exert influence on music you hear. An early an popular example for generative music was Mozart's musical dice game. Given a set of small sections of music, it was randomly chosen which part was played next.

1.6.2 Approaches

Genarative music (or algorithmic music) can be devided into the following subcategories [?].

Linguistic/Structural

Recomposition of analyzed songs, using grammars or stochastic processes.

Creative/Procedural

Randomly reorder pre-defined musical parts and play them.

Biological Emergent

(Simulation of) biological influences.

Interactive/Behavioural

No instruments - Recorded and filtered samples at the most. Synthesized music. Music generation fully controlled by user input/interaction. (see related work)

1.7 Possible User Interactions

Microsoft Kinect, acceleration sensor of a modern smartphone, etc.

1.8 Employed systems

1.8.1 VHDL

- Hardware components implemented in VHDL

1.8.2 ReconOS

- Developed at the University of Paderborn - Operating system running on a softcore alongside our VHDL components - Especially useful for integration of external devices (generative music)

1.8.3 Eclipse/GMF

- Synthesizer Editor developed as an Eclipse Plugin. - Will use GMF

2 Introduction

2.1 Introduction

"'Soundgates Interactive Music Synthesis on FPGAs"' is a project initiated by the Computer Engineering Group of the University of Paderborn, aiming to synthesise music on modern FPGAs. Furthermore, one or more users should use this interactive system, by means of manipulating the synthesised music with advanced sensors such as the acceleration sensor in modern smartphones.

2.2 Generative Music

2.2.1 Introduction

As pointed out in [**?**], there are many cultures where musicical experience is defined by people performing and people perceiving music. The only way to slightly exert influence on the performer's music is by cheering, shouting, etc. on a concert. The gap between performer and perceiver reaches its peak in the context of recorded music like CDs or MP3 files, where is no chance of manipulate the music. Actually, there is a rising trend for interacticing with music on your own or with a familiar social partner, like in the video game "'Guitar Hero"' [**?**, **?**]. Even without any musical knowledges or talent, it is more and more possible to interact, manipulate or even create music. Generative music combines the opportunities of making music without having knowledges of how to play an instrument and explicitly exert influence on music you hear. An early an popular example for generative music was Mozart's musical dice game. Given a set of small sections of music, it was randomly chosen which part was played next.

2.2.2 Approaches

Genarative music (or algorithmic music) can be devided into the following subcategories [**?**].

Linguistic/Structural

Recomposition of analyzed songs, using grammars or stochastic processes.

Creative/Procedural

Randomly reorder pre-defined musical parts and play them.

Biological Emergent

(Simulation of) biological influences.

Interactive/Behavioural

No instruments - Recorded and filtered samples at the most. Synthesized music. Music generation fully controlled by user input/interaction. (see related work)

2.3 Possible User Interactions

Microsoft Kinect, acceleration sensor of a modern smartphone, etc.

2.4 Structure

Chapter 2... Chapter 3... and Chapter 4!

3 Goals

We want to develop a graphical editor for the synthesizer development. The library of the editor offers components like sound generators, filters etc. which the designer can choose and connect to create an own patch with specific sounds. The developed patch can be exported as VHDL-code, which is synthesized and put on a FPGA. The user of the synthesizer (e.g. a musician) can connect a MIDI device or specific sensors to the FPGA to modify input values and to create different sounds.

To describe the functions, which we want to develop, in detail, we provide use case diagrams.

3.1 Editor for the designer

The designer can add components to the patch and remove them from the patch. He can connect components by drawing links between their ports. A component can be an atomic component like a sound generator, filter etc. The atomic components are stored within XML-libraries. Some of these components have static properties, which the designer can modify (e.g. the value of a constant-block). The other kind of components are composite components. The designer can add a composite component to the patch and fill it on his own with other components and links between them. He can add ports to a composite component and define their direction and constraints. We want to make it possible to export finished composite components as XML-files and to import existing ones to the editor such that different designers can exchange their components. Furthermore, the designer must define the interfaces of his patch. The interfaces are the points where the end-user interacts with the system. These can be MIDI-inpus, sensor values etc. It a patch is finished, it can be validated. The program tests if all constraints are fulfilled (e.g. if all ports are connected and so on). The designer can export his patch as XML-file and build it, which means that he can generate VHDL-code out of the XML-file.

3.2 Simulator

We want to develop a software simulator for a patch such the designer und the user can test the sound of the patch before putting it on the FPGA. A designer or an end-user must be able to import a patch to the simulator and to start the simulation. Then he can pause or stop the simulation. An extra feature we want to implement is that the simulation can be recorded.

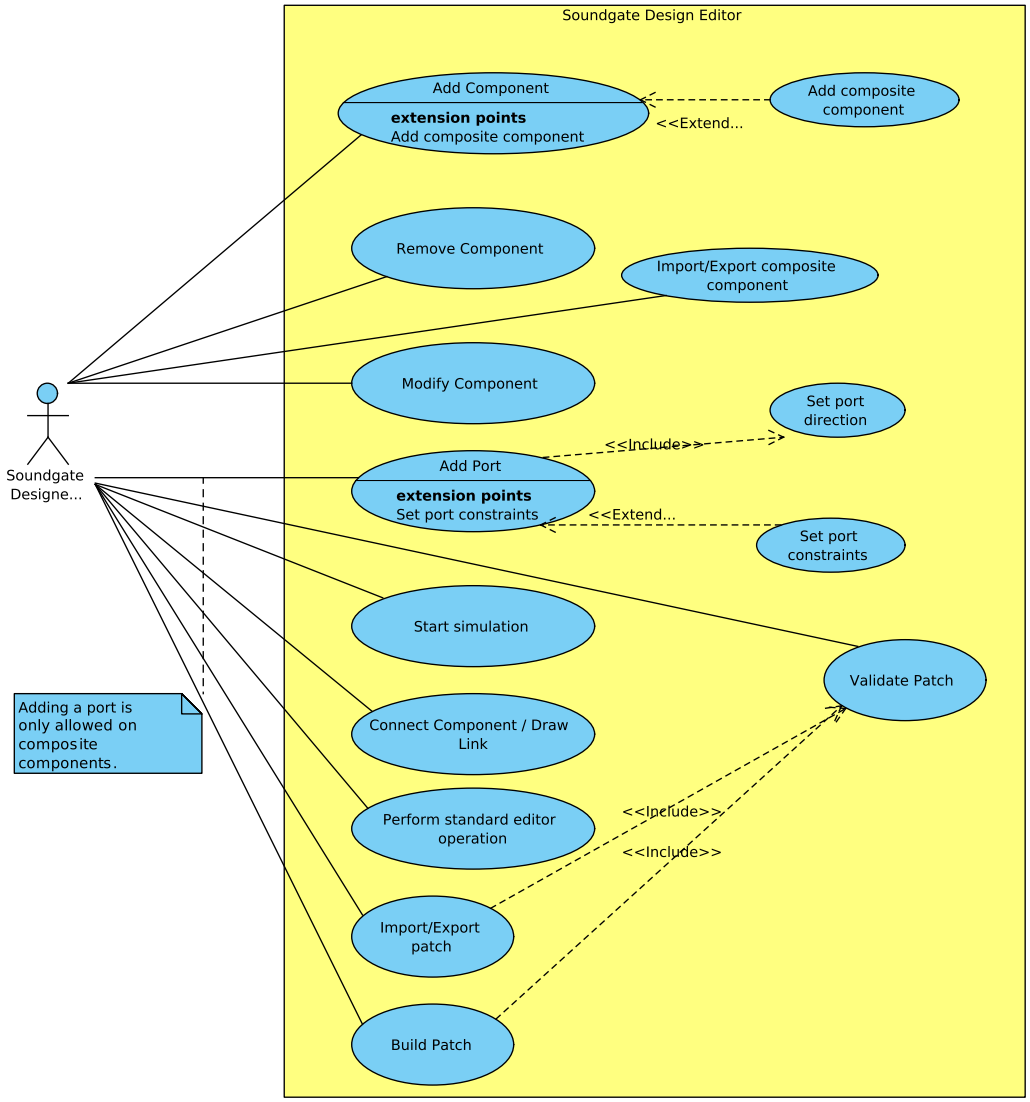

Figure 3.1: Use case diagram of the soundgate designer (editor)

3.3 End-user product

As mentioned before, a patch created in the editor can be exported as XML-file which is used to generate VHDL-code. This VHDL-code is synthesized and put on a FPGA. The end-user (e.g. a musician or a performer) maps sensor values and other inputs to the interfaces defined by the designer (see ..). He starts the session by pushing a button. During the session he performs some sensor actions or actions with other devices to create input values to the system. An extra feature we want to develop is that a session can be recorded.

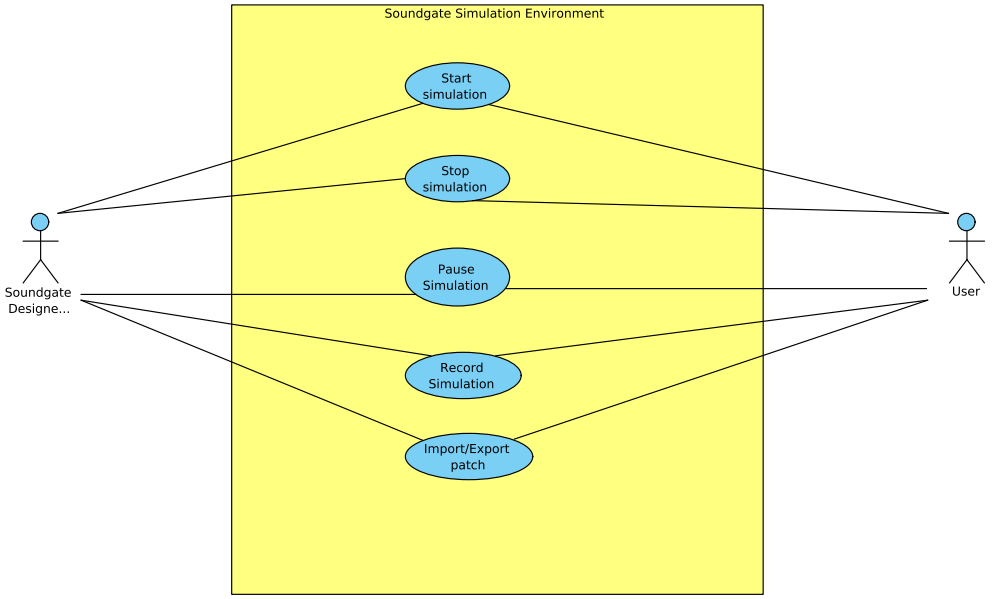

Figure 3.2: Use case diagram of the soundgate simulation environment

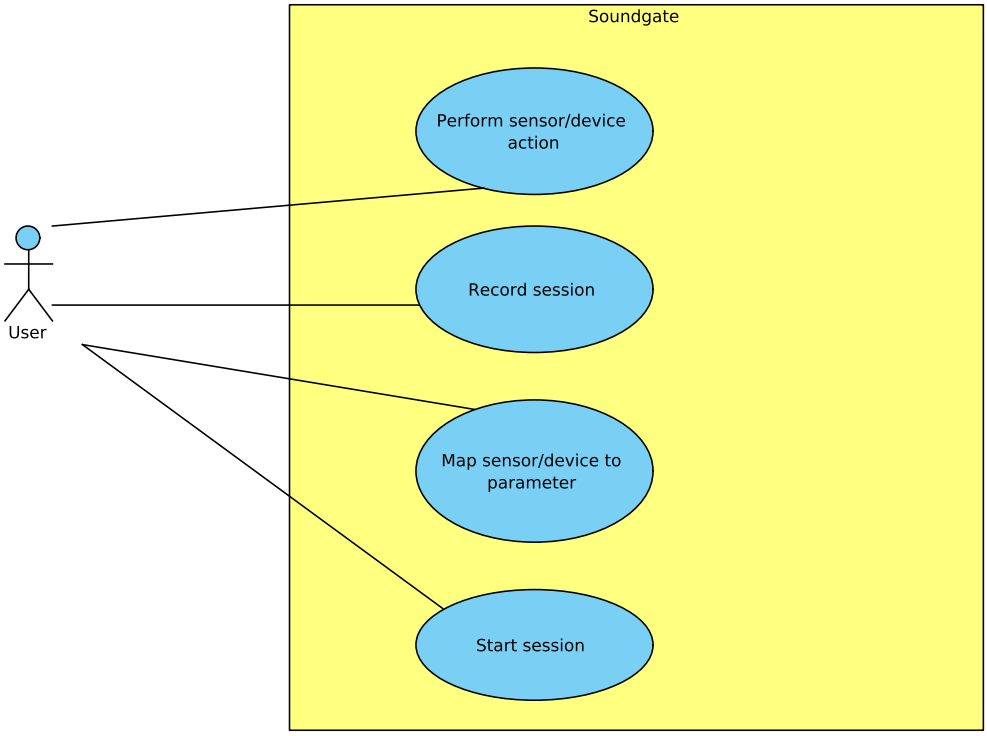

Figure 3.3: Use case diagram of the soundgate user interface

3.3.1 Input

We plan to use smartphones as sensor devices. For this purpose we want to develop a special smartphone app. Furthermore the end-user should be able to use MIDI devices.

3.4 Component library

We plan to provide following components:

- Generators:
  - Sinus
  - Sawtooth
- Filters:
  - Ramp
  - Low pass
  - Delay
- Arithmetic:
  - Addition
  - Multiplication
  - Equals
- Mixer
- Sources:
  - Constant
- Sinks:
  - Sound output
- MIDI:
  - MIDI note to frequency converter
  - MIDI scheduler

4 Related Work

4.1 Cycling '74 Max

Aufbau, Elements, User Interaction

4.2 Reactable

Aufbau, Elements, User Interaction

5 Organization

6 Workplan