

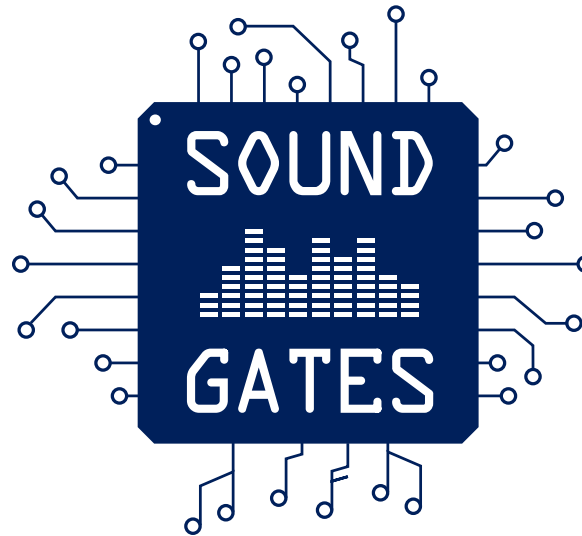
Projectplan



Outline

- Introduction
- Generative music
- Soundgates
- Technologies
- Workplan





Introduction



Music

- **Traditional:**

Musicians perform and people perceive music

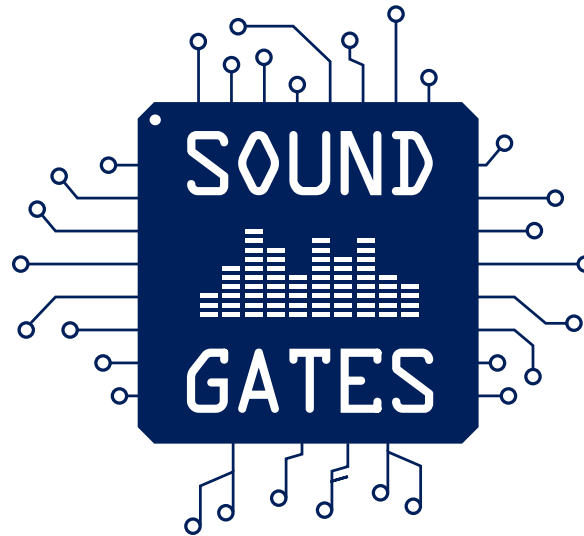
- **Trend:**

Interact with music (even without knowledge)

- Cheering and shouting at a concert
- Guitar Hero, Rockband, DJ Hero, Singstar, ...

2 Level of sound generation

- **Goal:**
Generate music in Hardware on a FPGA
- **Level 1:**
Musician connects components to generate sounds and melodies
- **Level 2:**
User interacts with system at runtime to modify the output
 - Motion Sensors
 - 3D depth camera (i.e. Kinect)



Generative music



Approaches to generative music

- Creative / Procedural
- Interactive / Behavioural



Approaches to generative music

- Creative / Procedural
- Interactive / Behavioural

Creative / Procedural

- **Goal:**
Generate music from precomposed options
- Musician writes song which consists of different parts
 - parts are exchangeable and randomly played
 - generates every time a new song
- ie. “Mozarts Musical Dicegame”
 - next played part was randomly chosen by rolling a dice

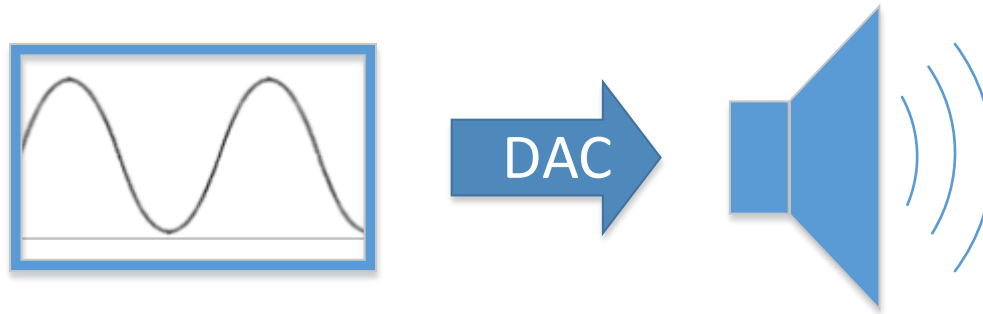
Approaches to generative music

- Creative / Procedural
- Interactive / Behavioural



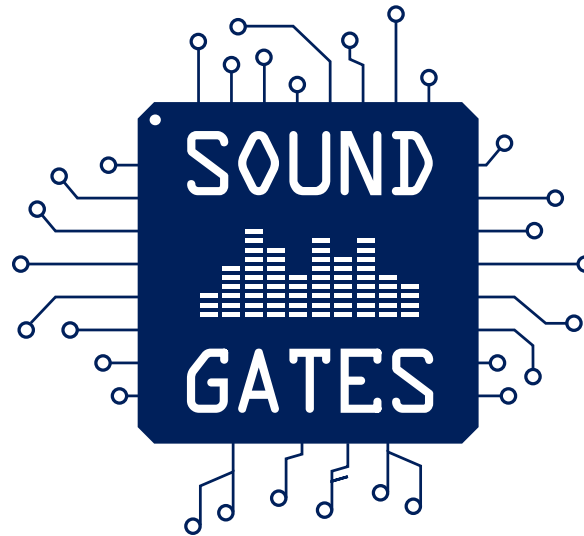
Generate Sound on a digital System

- Simple synthesizer



Interactive / Behavioural

- Results from processes without discernable musical inputs
 - uses:
synthesized music
- Process of music generation/manipulation is fully controlled by user input and interaction
 - input modified with sensors



Soundgates



Soundgates

- Editor
- Simulator
- COSMIC



Soundgates

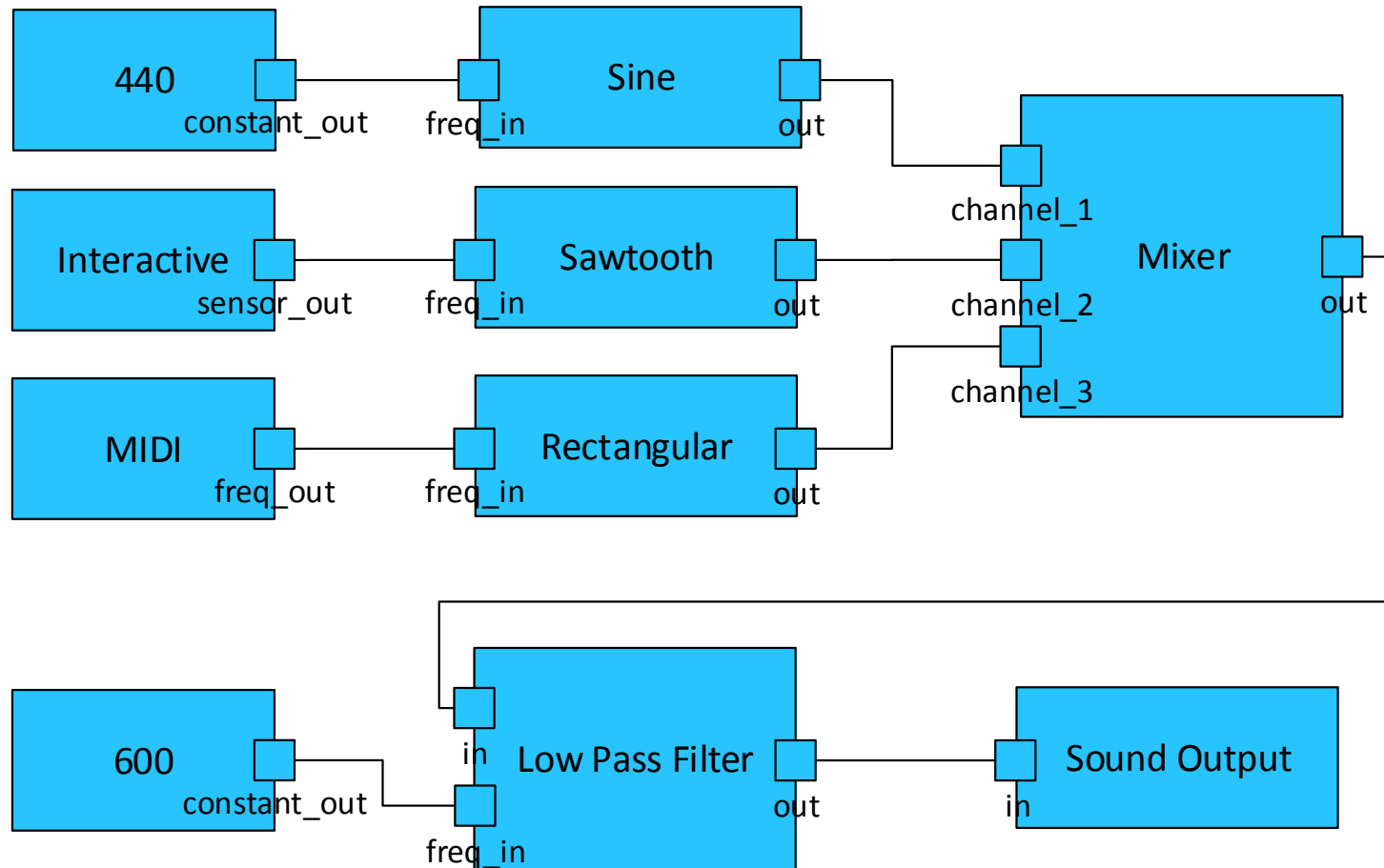
- Editor
- Simulator
- COSMIC



Editor

- Musician builds/loads a patch
 - consists of sound-components and connections
- Sound-components
 - wave generators (sine, sawtooth, rectangular)
 - arithmetic functions (i.e. addition, multiplication)
 - filters (i.e. low pass)
 - mixers
 - composite sound components

Example patch



Editor functions

- Make sound component public
 - possible to modify at runtime with sensors
- Export patch to VHDL code
 - and synthesize it to Bitstream
- Validate patch
 - constraints
 - i.e. every port has an input

Soundgates

- Editor
- Simulator
- COSMIC



Simulator

- **Problem:**

Testing the output is not possible until VHDL code is synthesized

- **Solution:**

- Test the developed system on PC
 - Every sound component will be implemented in SW & HW

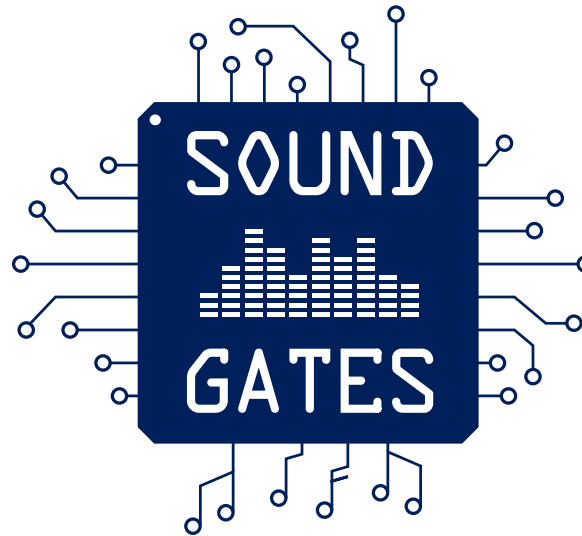
Soundgates

- Editor
- Simulator
- COSMIC



COSMIC

- **Computer Scientists Making Music**
- The generated Bitstream is put on an FPGA
- Performer maps sensors to interfaces
 - starts session by pushing a button
- Creates input values with sensors
 - music will be generated / modified



Technologies



Graphical Modeling Framework (GMF)

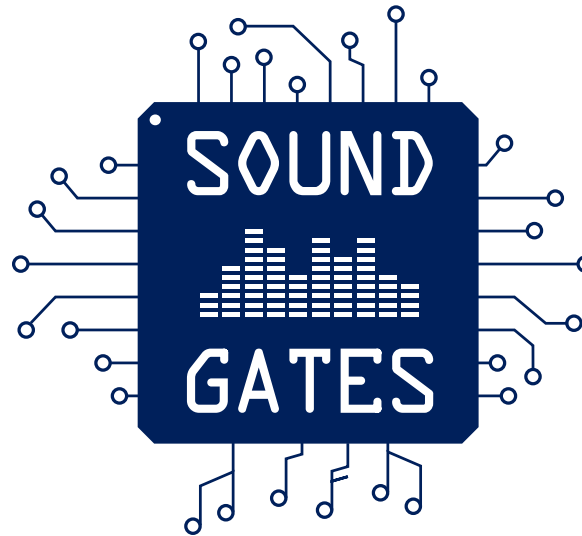
- A “Model Driven Software Development” approach for graphical editors
- Specify Metamodel and generate software
 - parts and rules which are needed to create valid patches
- **Used for:**
Create graphical editor to build patches

ReconOS

- Model applications using software and hardware threads on an FPGA
 - Posix-like abstraction (mailboxes, semaphores, ...)
 - communication between threads abstracted by method calls
- **Used for:**
Sensor input comes via IPC and is processed in software
 - modifies parameters of HW

Open Sound Control (OSC)

- Message based communication protocol
 - developed for communication between computer systems and sound synthesizers
- OSC message modifies parameters
 - i.e.
/synthesizer1/oscillators/sine/freq “int32” 440
- Independent of transport protocol
- **Used for:**
Sensors will send OSC messages to FPGA system to modify parameters



Workplan



Agile inspired development process

- Split into Hardware- and Software Groups
 - “Cross testing”
- 5 milestones
 - each consists of a set of tasks
 - approximately five to six weeks per milestone
- “Github” for versioning and sharing of code
- “Redmine” to represent milestones and tracking of tasks and bugs
- Functional system at the end of every milestone

Milestones

1. Prototyping infrastructure / environment
 - fundamental infrastructure is prototyped
 - no direct communication between them

2. Prototype of a digital synthesizer
 - basic digital synthesizer can be modeled with the editor
 - transform patch to HDL description

Milestones

3. Polishing environment

- every planned component is implemented
- create Android application to stream sensor data to the COSMIC system

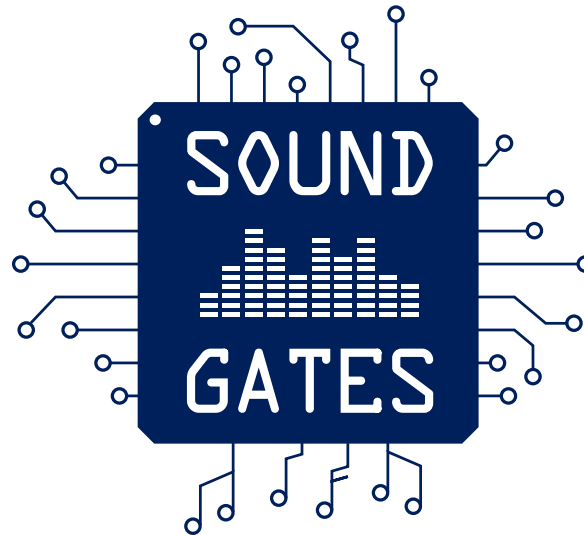
4. System integration and benchmarking

- evaluate system limits

Milestones

5. Documentation, Testing, Presentation
 - polishing phase

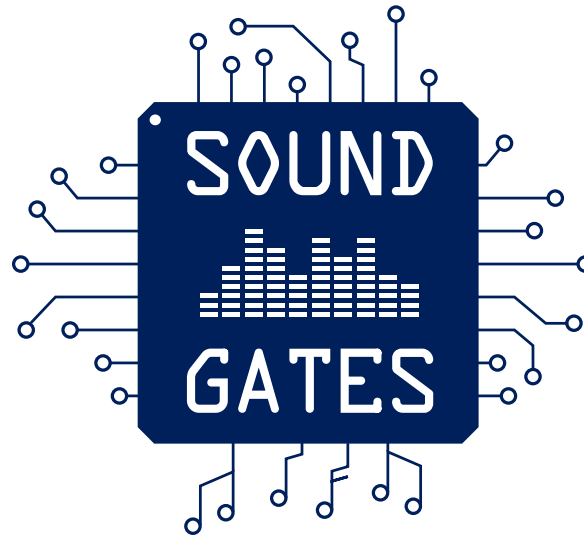




Thank you for your attention –

Any comments, hints, questions?





Backup



Open Sound Control (OSC) - Datatypes

- int32
- float32
- OSCString
- OSCTimetag
- OSCBlob

Open Sound Control (OSC)

- Advantage:
 - higher speed than midi
 - larger datatypes and float values
 - can be send via ethernet
 - “Internet-Jam”
- Disadvantage:
 - namespace does not follow any standard