Dokumentation von Tests und Verhalten

Julian Hatzky

August 10, 2019

Contents

Part I

Audio Signal: speaker output

The first major test that is implemented is to forward the incoming signal directly to the output. That enables the possibility to plugin a speaker so that it is possible to control the Audio signal by listening and comparing. However keep in mind that the human ear can detect differences in audio signals only to a certain degree.

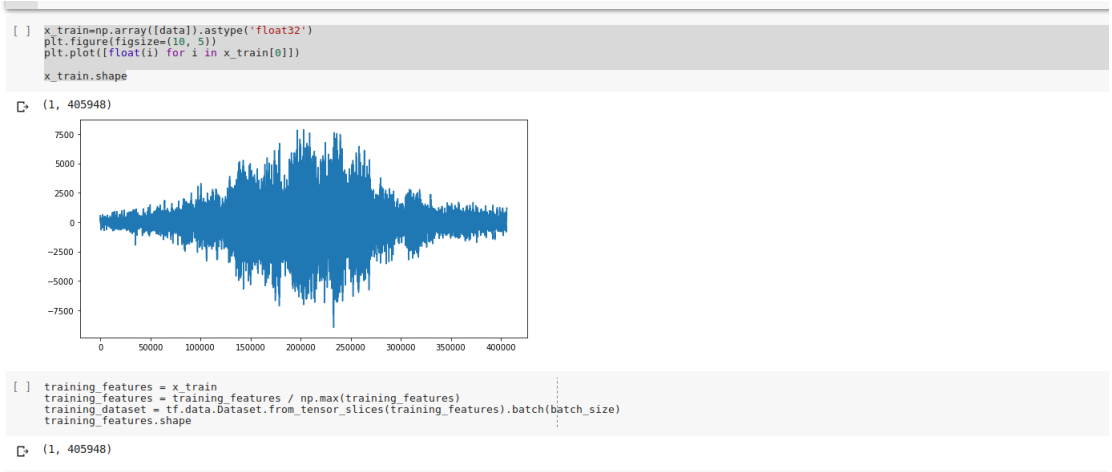
Part II

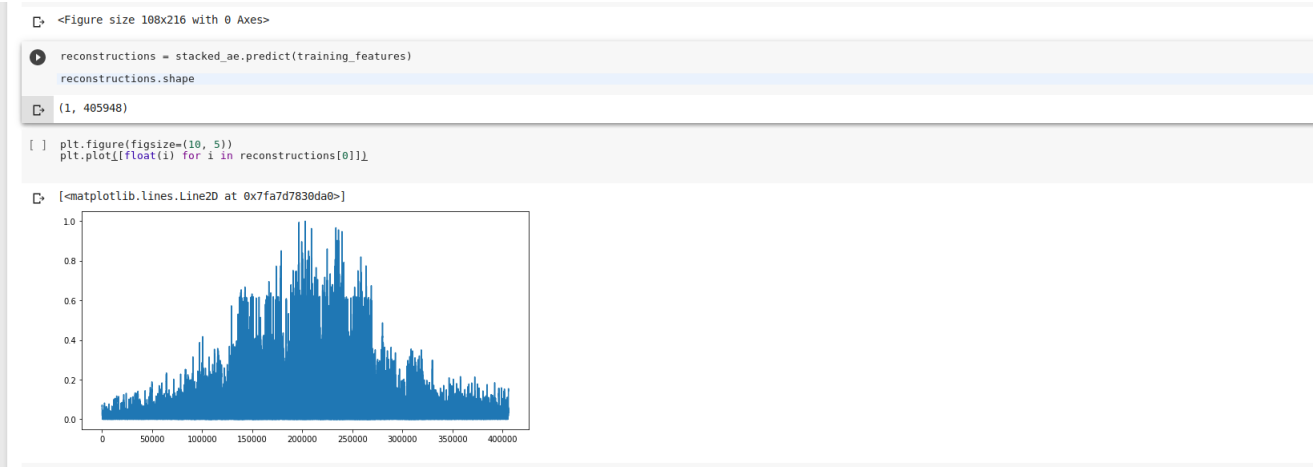
Test of the algorithm

Another step that is also part of the testing procedure was to evaluate the Neural Networks algorithm on a computer first. This leads to a proof whether the algorithm and architecture that was choosen - the AutoEncoder - is able to detect certain patterns and filter out random noise.
For the evaluation Tensorflow 2.0. The whole python script can be found in the drive folder as a google colab document and under the code section.

The proof-of-concept shows that it is possibel to create a more accuarate representation, of the important parts of the signal, out of the input audio signal.

The next two figures are showing the original audio data (upper picture) and the latent pattern representation (bottom picture).

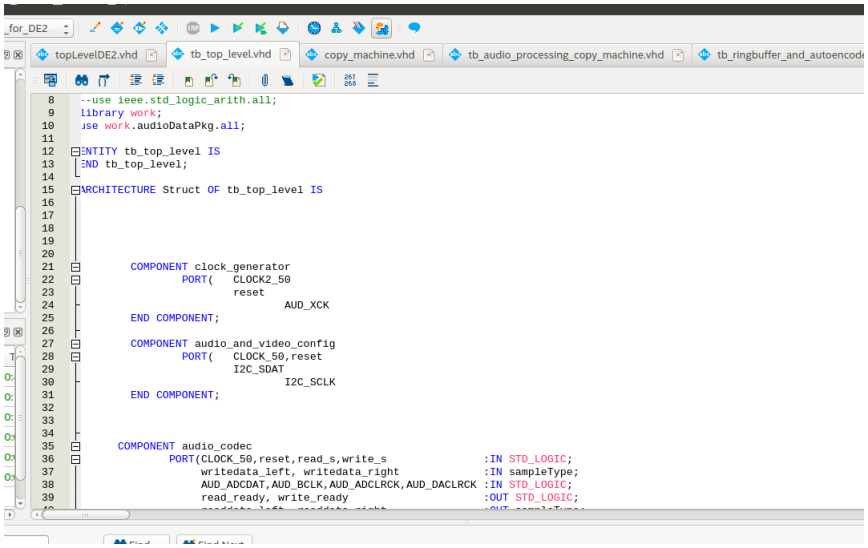


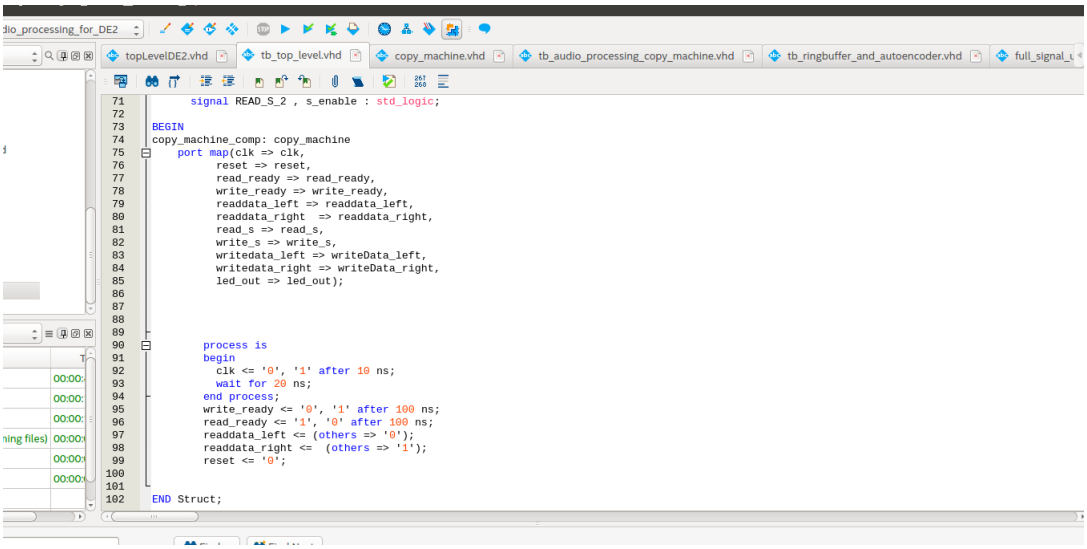


Part III

Testbenches

Testbenches are extremly important when creating high quality code with Quartus. Since Quartus is not easy to debug, Testbenches are one of the fastest and most straight-forward ways to test either specific components of your system or the architecture as a whole by running them in a simulation. In this section a desciption of the Testbenches that were used during the project is given as well as further explanations and ideas.
- Snippets of the top-level Testbench can be seen on the pictures bellow.





Testbench Top-Level Architecture

The top-level Testbench is the test of the system as a whole. It is the last instance before the system can be tested on a real FPGA. A clock signal is given that switches every 10 nanoseconds within the Testbench's process. Furthermore instances of all the top-level components are created and connected to Testbench signals so that it is also possible to read inner signals of the system within the simulation process. However the main aim of this Testbench is to look onto the whole system as a black box, give some well-defined and necessary signals to the inputs and check if the output matches the expecations.

For more unit level analysis the following described Testbenches are likely to be better candidates.

Testbench Copy-Machine

In this Testbench an the copy-machine is instantiated. Hence the other components are Altera IP they are expected to behave correctly per default. That is why this Testbench focuses on the functionality of the copy-machine. Again it is possible to look at the copy-machine as a black-box and observe the output signals while changing the input signals and see if the behaviour matches the expectation. However the most relevant signals of the copy-machines inner circles are again connected to Testbench signals so an observation during the simulation is possible.

Testbench Ringbuffer - Denoising AutoEncoder

The most complex part of the whole system is the processing of data between the component Ringbuffer and the component Denoising AutoEncoder. To ease the analysis of this key route another Testbench has been created. Within this system every single Signal between all those components is measured. This Testbench is crucial during development to get a quick understanding where certain error occuring. The picture below shows the relevant part that is tested with all its signals.