ErUM-WAVE
Anticipation of 3-dimensional wave fields

Alexander Bauer, Michael Bussmann, Anjali Dhabu, Waleed Esmail, Dirk Gajewski, Oliver Gerberding, Celine Hadziioannou, Conny Hammer, Markus Hoffmann, Katharina Isleif, Alexander Kappes, Julian Rautenberg, Stuart Russell, Holger Schlarb, Morvarid Saki, Achim Stahl, Jochen Steinmann, Christine Thomas

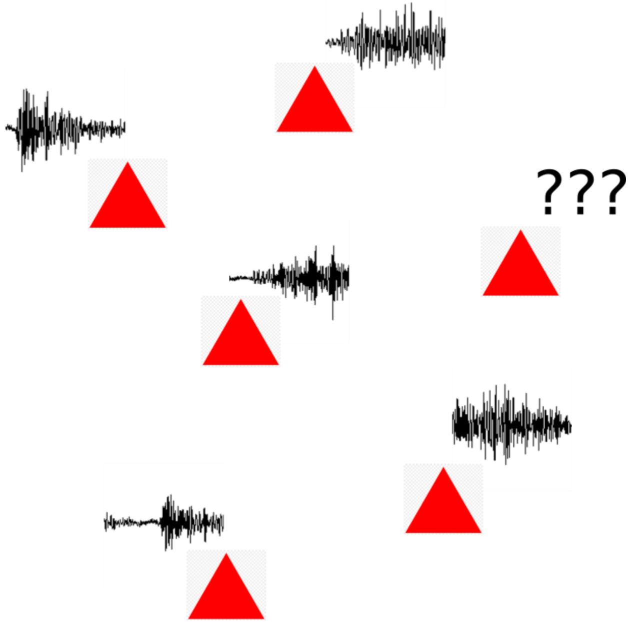
Outline

- **What is ErUM-WAVE ?**
  - Overview

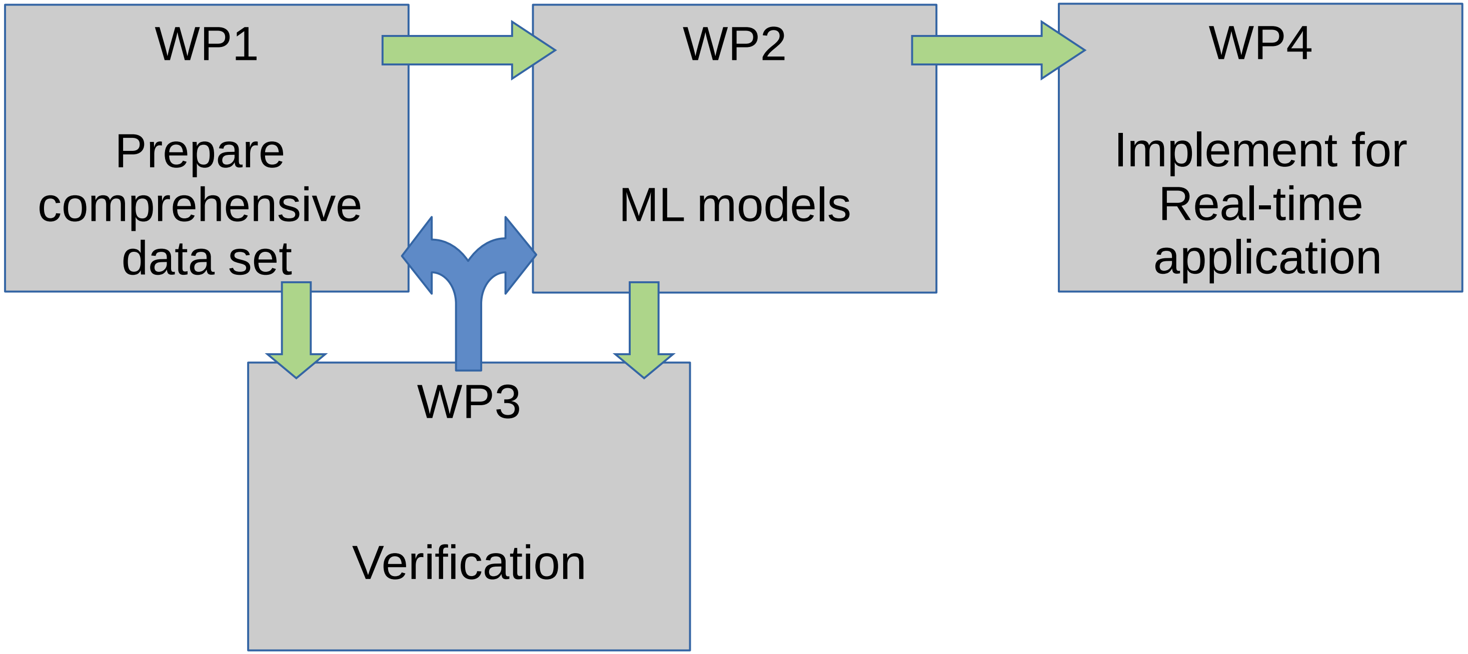
- **Our Test-Setup in Aachen**
  - Active Noise Mitigation for a Michelson Interferometer

Aim - Reconstruct wave field from measured data of network of point sensors, predict further propagation

- Focus
  - seismic waves
  - anticipate subsurface motion at different scales
- Goal
  - active compensation of vibrations
  - applicable to real situations, local properties of medium have to be taken into account
- Application
  - PETRA-III/IV, XFEL, gravitational wave detectors
  - radio wave reconstruction in AUGER

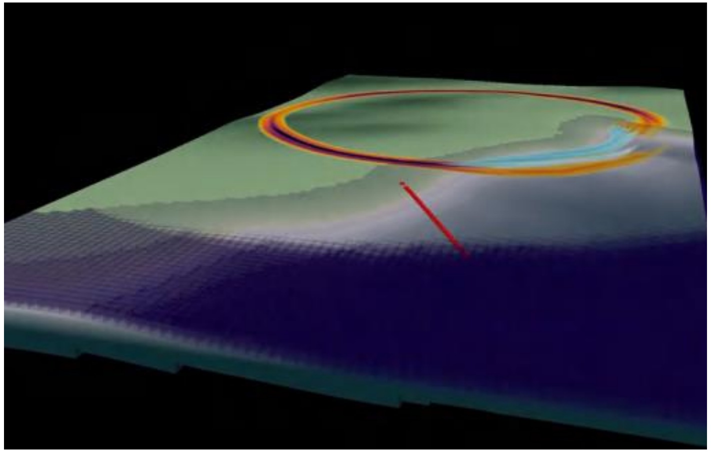


How to


WP1
Prepare comprehensive data set
WP2
ML models
WP4
Implement for Real-time application
WP3
Verification

WP1 - Simulate subsurface structure and propagating seismic wave field

- **Create model of subsurface structure** from measured seismic data
  - ➢ large/medium/small scales
  - ➢ simplified structural model
  - ⟶ reflections and anisotropy
  - ➢ Field measurements in region of ET
- **Simulate seismic wave field** by forward modelling
  - ➢ Classical seismic approach
  - ➢ Surrogate models (grid-free, high-quality interpolation of high-dimensional parameter spaces)
- Provide comprehensive training data set for WP2

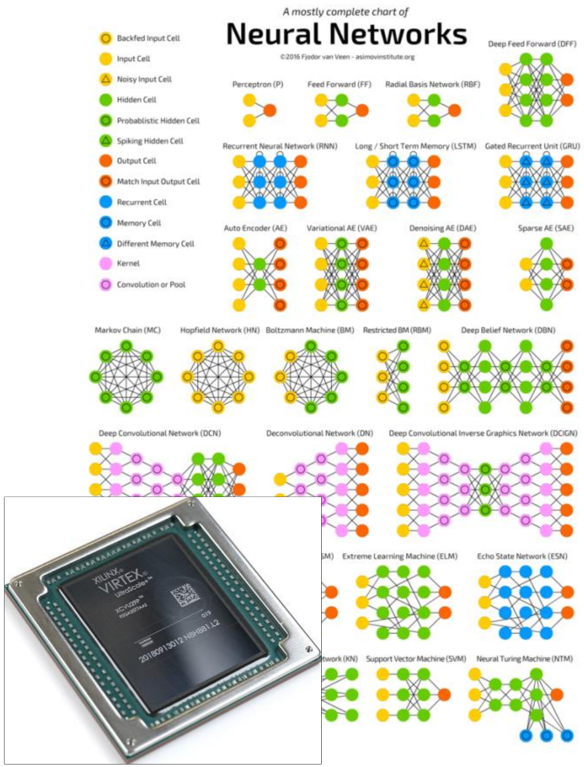


(mondaic.com)

WP2 – Method development, networks and AI training

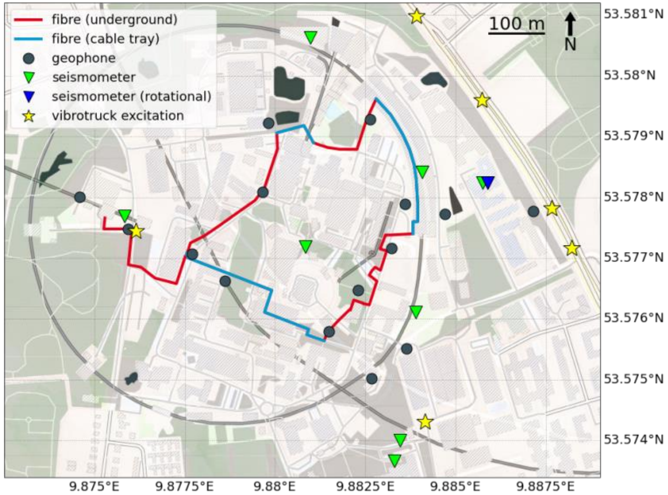
- Test different neural network architecture for 3D wave propagation
  - **Surrogate models:** enable grid-free, high-quality interpolation of high-dimensional parameter spaces

    → generate artificial datasets to train other architectures (WP1)
  - **CNNs:** first tests on recorded seismic traces show promising results, (130 earthquakes recorded at 3C-stations in northern Germany)
  - **Transformer models:** self-attention mechanism, long time-series
  - ...
- An essential goal is to develop architectures that can be implemented in FPGAs for real-time predictions (WP4)



WP3 - Verification

- **DESY**
  - WAVE network: focus on **small scale**, anthropogenic and technical noise
  - Application to PETRA-III and EuXFEL: vibrations of base plate, tunnel segments, ...



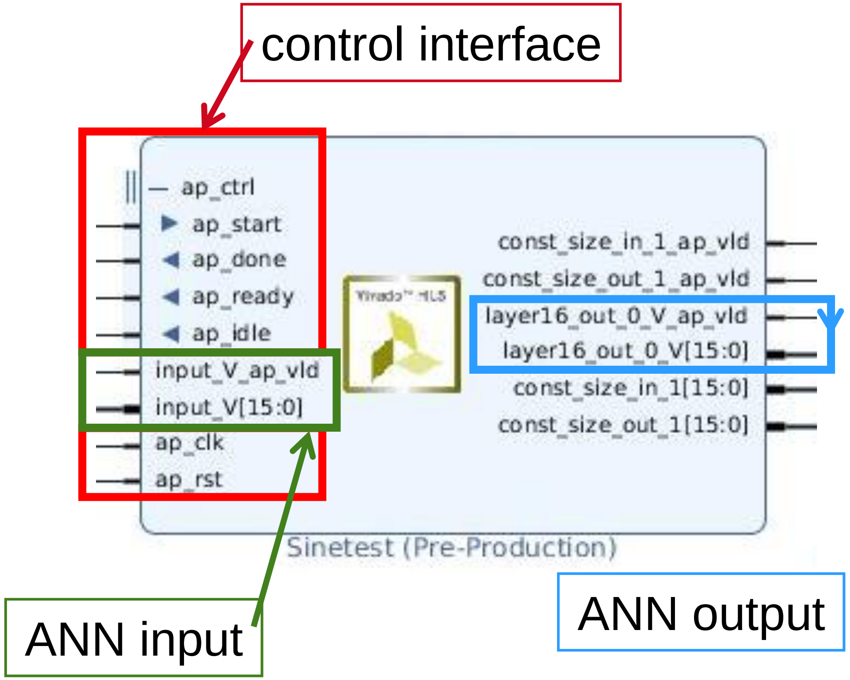
- **ET**
  - Focus on **large scale**: local, regional, global earthquakes (surface waves, P-/S-waves)
  - Simulated data of GRSN to predict at target station close to ET

- **AUGER**
  - Measured data: sufficient number of events/stations
  - Simulated data: higher spatial resolution

ErUM-WAVE

WP4 – Getting an ANN inside a FPGA (XILINX)

- Quantisation
  - **No floats**: in-/outputs, weights need to be quantisized
  - **Post Training Quantisation**: Training using floats, quantisation after full training
  - **Quantisation Aware Training**: Training using fixed point numbers
- Pruning
  - Remove small weights
  - Reduce parameters/mathematical operations
  - done after main training (re-training)
- Converting the model to HLS4ML



Summary

- Data-driven method for reconstruction of the full 3-dimensional seismic wavefield.
- Real-time prediction at different scales: regional and local ground motion, building, measuring plattform, sensor.
- Provide input for feedback-systems for noise suppression due to seismic disturbances or seismic noise cancellation in post-processing.
- Applications
  - “Noise-free-labs”.
  - Reconstructed seismic wavefields for exploration of subsurface to characterize geo risks.
  - Seismic risk assessment.

Active Noise Mitigation

our sandbox for real time ANN implementations

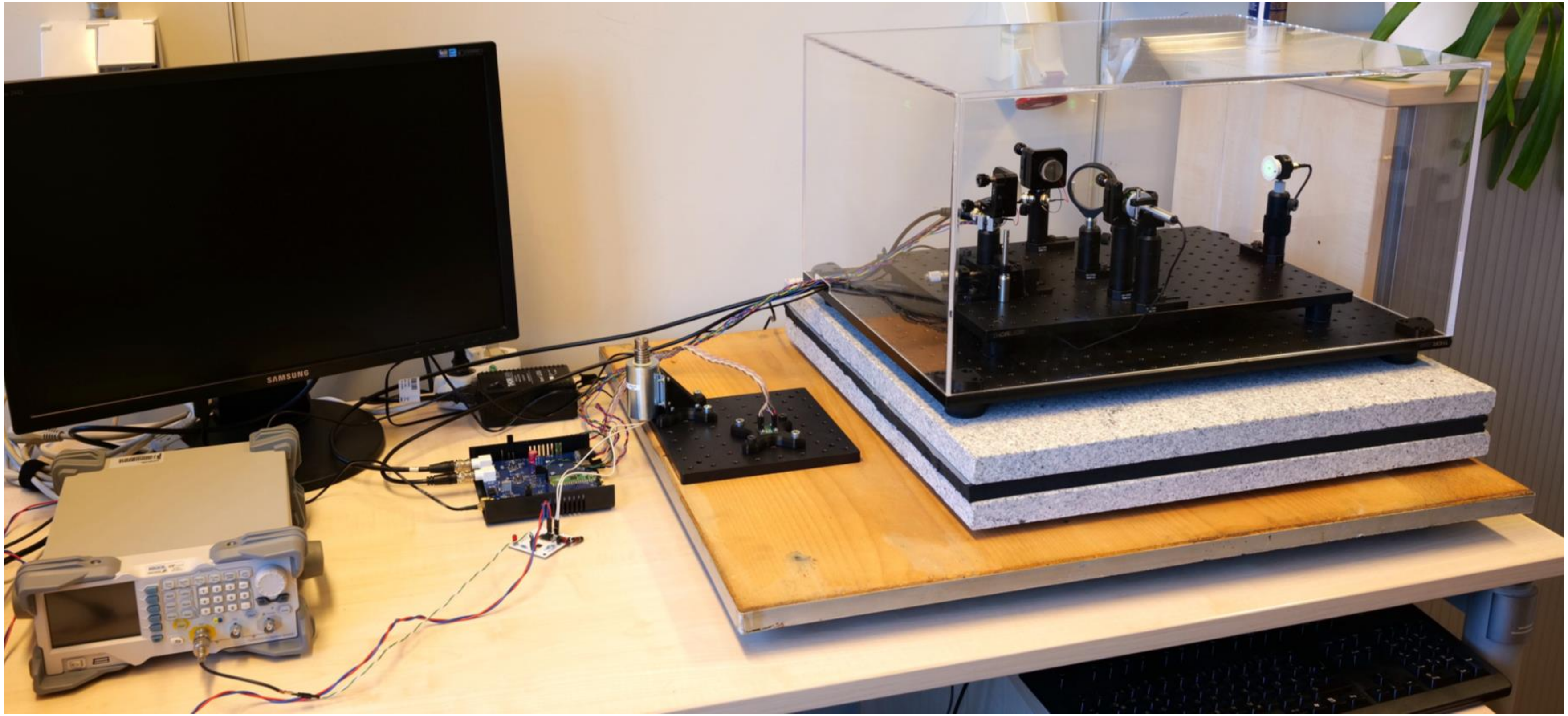
Quite an „easy“ task –

once the toolchain is established, more complex networks can be tested

Together with the AC ET group: Markus Bachlechner, Tim Kuhlbusch

ErUM-WAVE

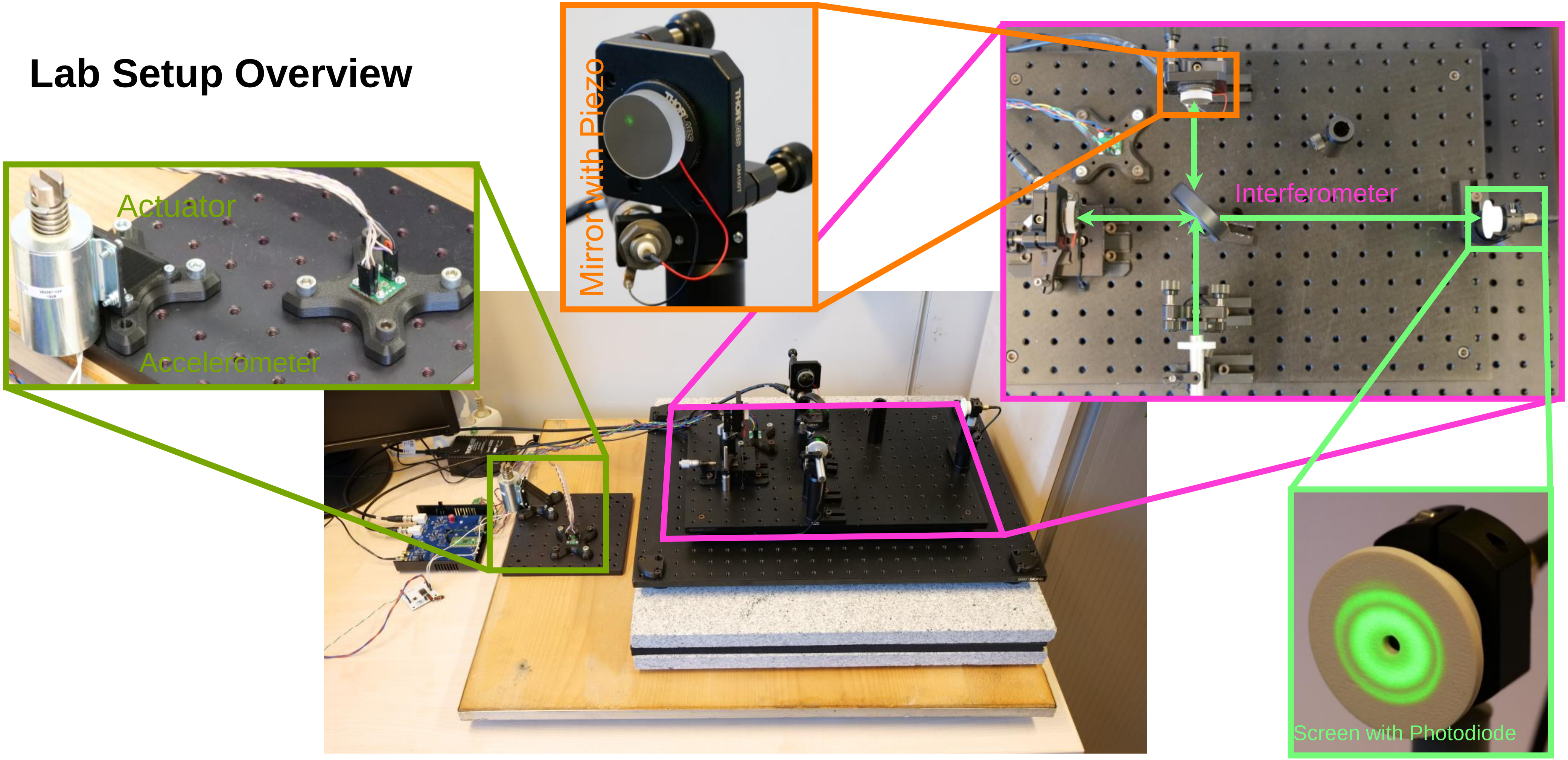
Lab Setup Overview



Goal: Online noise mitigation of demonstrator interferometer based on real data

ErUM-WAVE

Lab Setup Overview

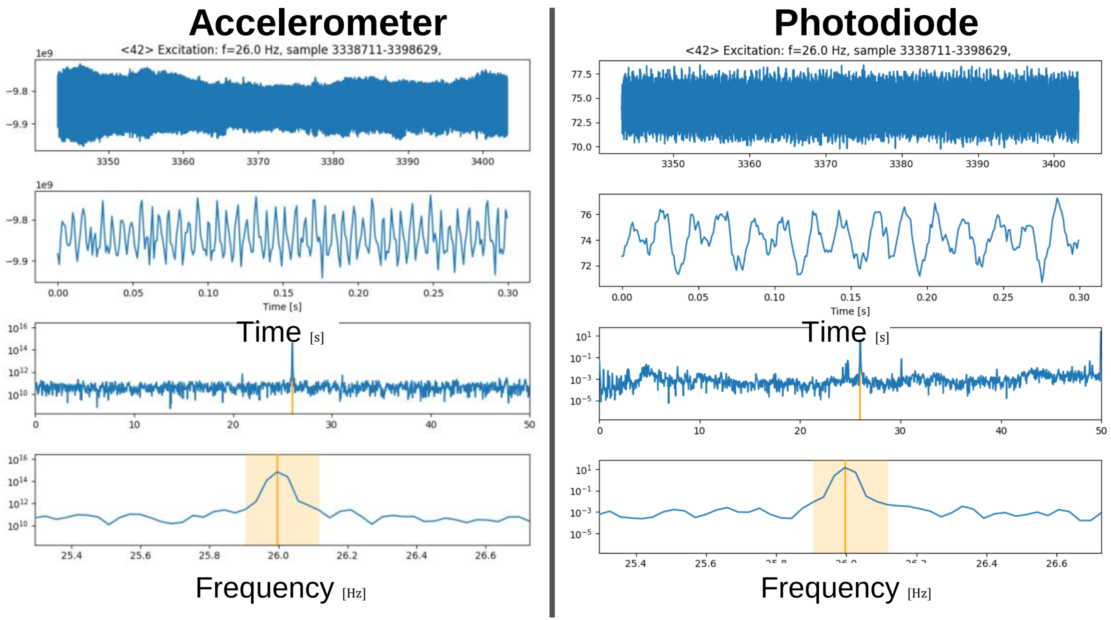


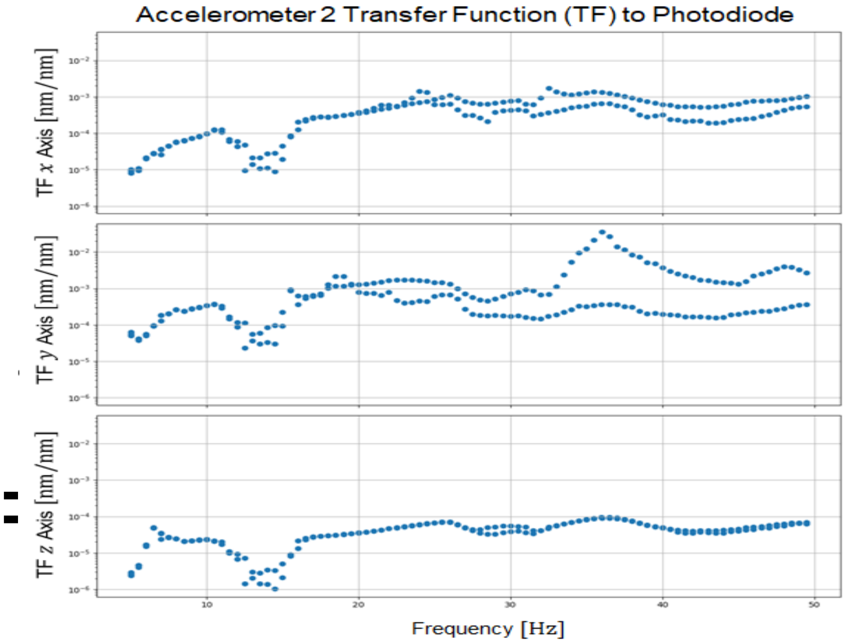
ErUM-WAVE

Characterization and Frequency Response

Determination of the Transfer function:

- Sinusoidal excitation with fixed frequency
- Determine measured amplitude at given frequency
- Scan through frequency band





ErUM-WAVE

Noise Mitigation Neural Network

Mitigation Principle and Optimization

Input:

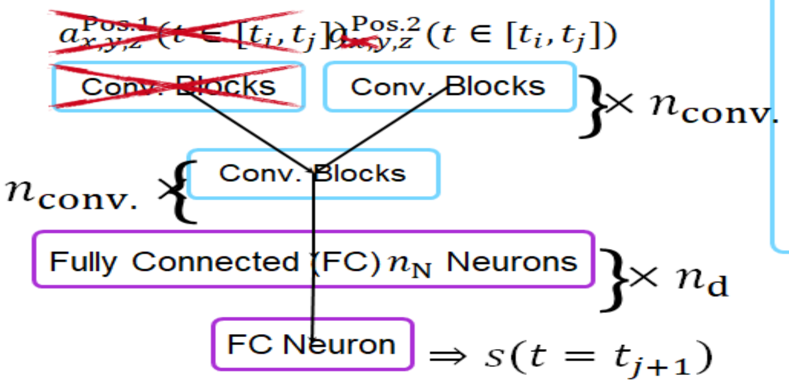
- Measurement accelerometers $a_k^{\text{Pos.1/2}}(t)$ *with* $k \in \{x, y, z\}$
- Window of duration T ($\sim 100\text{ms}$) from t_i to t_j
- White noise excited at $10 - 30\text{Hz}$

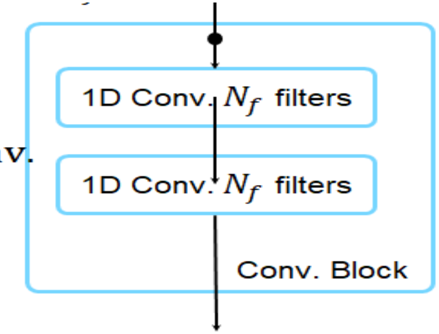
Output:

- Prediction for photodiode signal $s(t)$ at time t_{j+1}

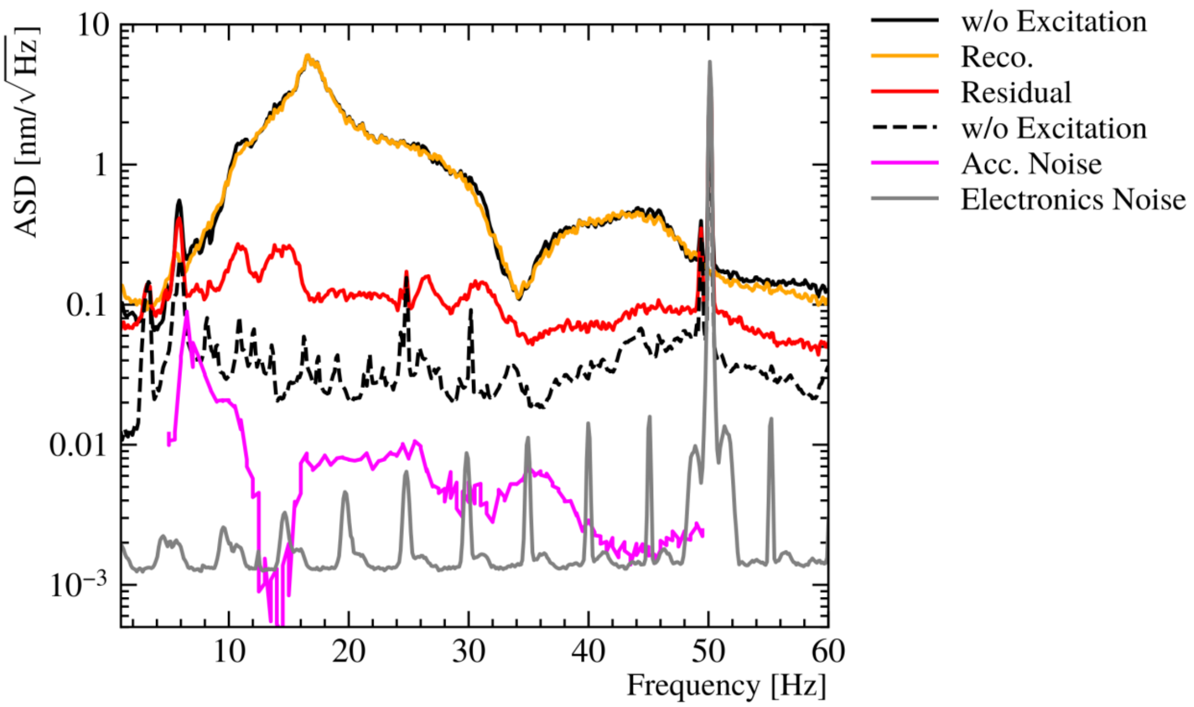
Porting to FPGA:

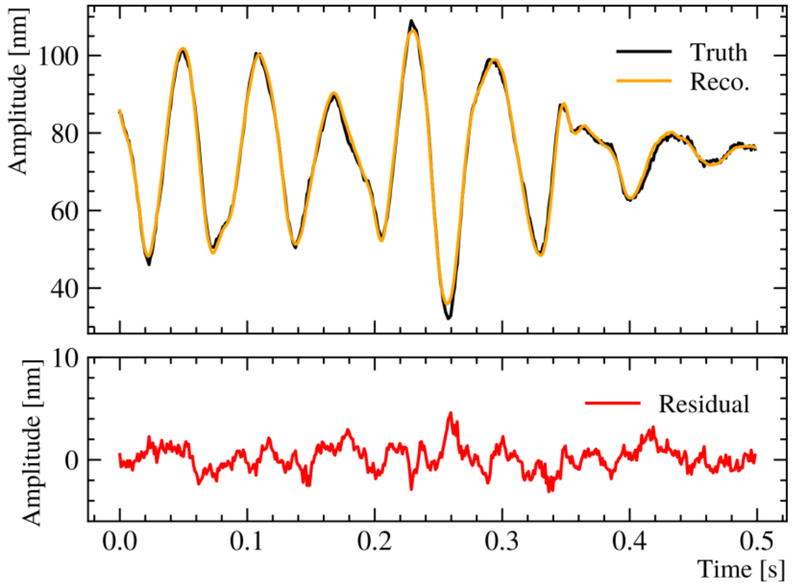
- Limited storage and number of operations per clock cycle
  - Reduce size of model ⇒ **pruning**
  - (No floats ⇒ **quantization** of parameters)
- Performance of pruned network depends on initial state
  - Find “optimal” configuration ⇒ Hyperparameter optimization
  - Reduce to 2000 non-zero parameters





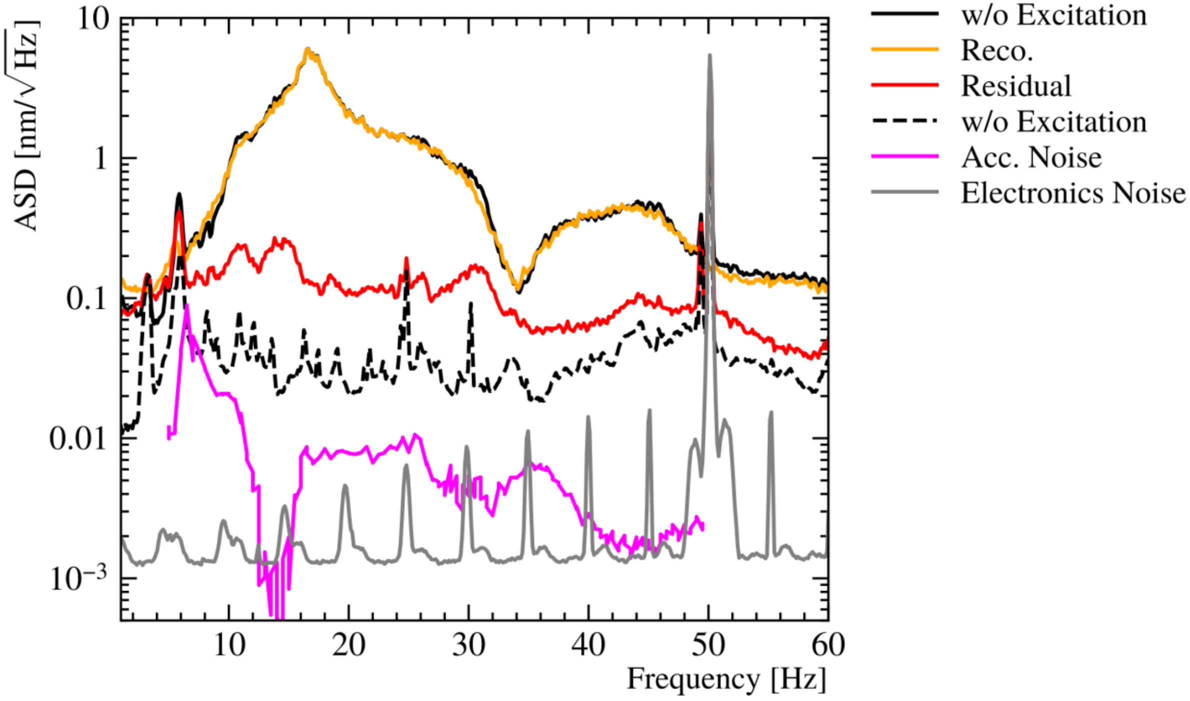
Performance Evaluation - BEFORE Pruning

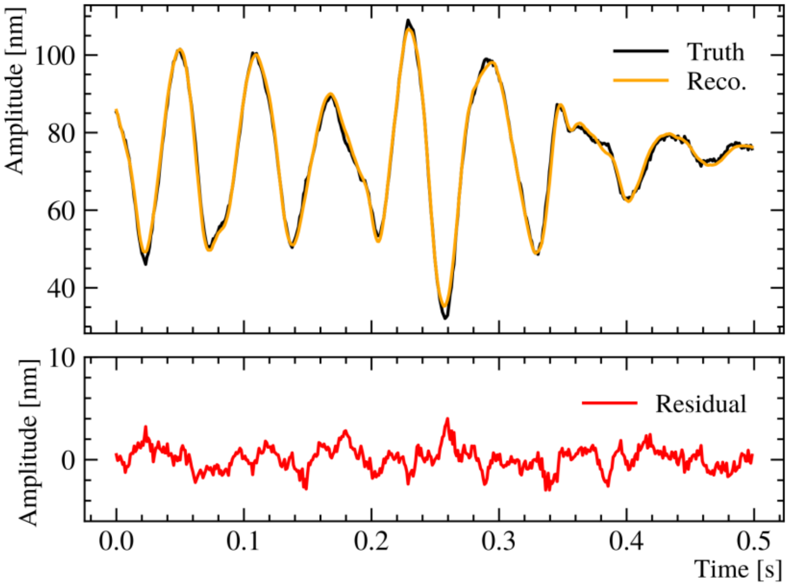




Metric	w/o pruning
ε_{tot}	95.79 %
$\varepsilon_{\text{w/o 50 Hz}}$	99.15 %

Performance Evaluation - AFTER Pruning



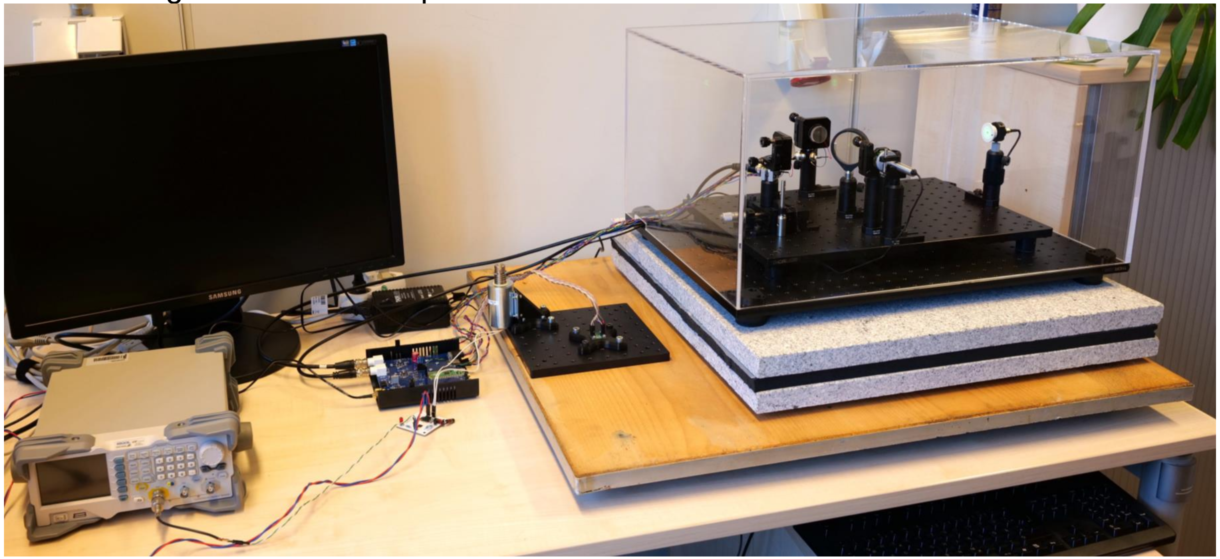


Metric	w/o pruning	w/ pruning
ε_{tot}	95.79 %	95.78 %
$\varepsilon_{w/o\ 50\,Hz}$	99.15 %	99.14 %

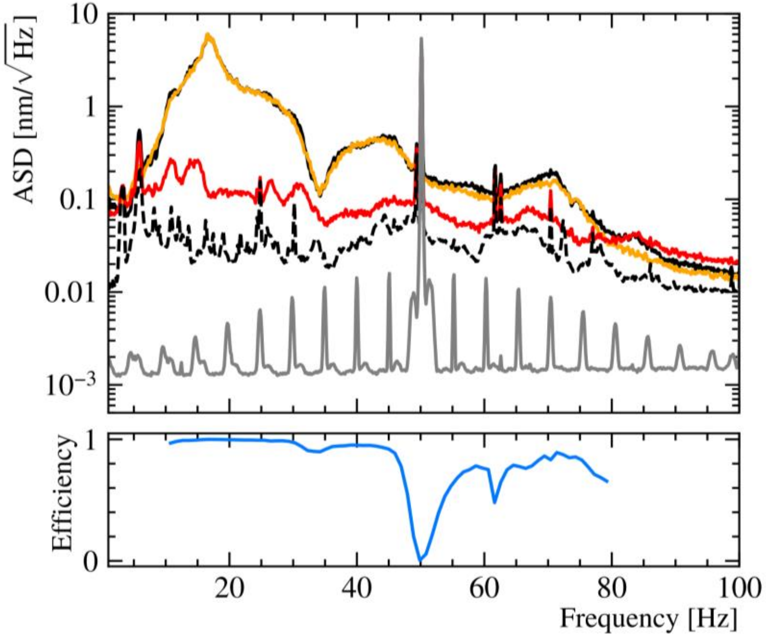
Summary Python Toolchain

Summary:

- Build setup to develop model independent online (not yet) noise mitigation based on real measurements
- High cancelation efficiency of 99.14%
- Even though 99.1% of 215k parameters are removed



Next: FPGA implementation



The dream of a physicist …

Unlimited parameters – very good performance

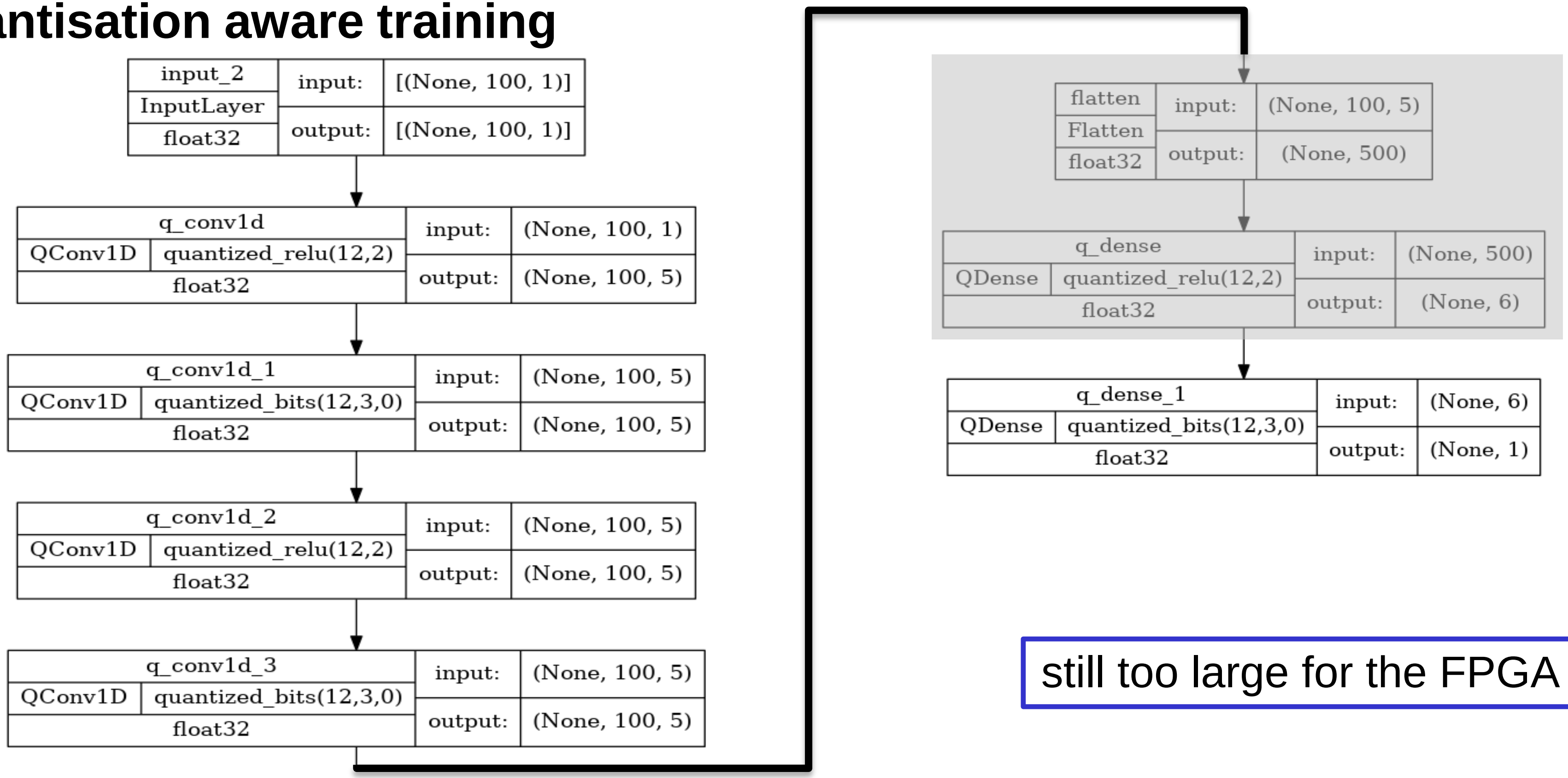
- **BUT: if the network should run on the FPGA → there are some limitation**
  1. No floats
     → use fixed point instead
  2. (very) limited storage for weights
     → use low bit width
  3. (very) limited amount of multiplications (240 DSP slices)
     → reduce complexity or use DSP more often
  4. Not all types of layers are supported (yet?)
     → use alternative structures and layers
  5. Special treatment of input and output format (in/output data scaled to -1 to +1)
     raw bits of a sensor match the fixedpoint representation of the training data

Quantisation aware training

Use the quantisation during training

New „reduced“ network

Quantisation aware training



still too large for the FPGA

Number of operations

- Still > 27k multiplications
- CONV1D are implemented using „*stream*“ implementation with less DSP per layer

```
Number of operations in model:
    q_conv1d                           : 1500   (smult_12_8)
    q_conv1d_1                         : 7500   (smult_12_12)
    q_conv1d_2                         : 7500   (smult_12_12)
    q_conv1d_3                         : 7500   (smult_12_12)
    q_dense                            : 3000   (smult_12_12)
    q_dense_1                          : 6      (smult_12_12)

Number of operation types in model:
    smult_12_12                        : 25506
    smult_12_8                         : 1500

Weight profiling:
    q_conv1d_weights                    : 15     (12-bit unit)
    q_conv1d_bias                       : 5      (12-bit unit)
    q_conv1d_1_weights                  : 75     (12-bit unit)
    q_conv1d_1_bias                     : 0      (12-bit unit)
    q_conv1d_2_weights                  : 75     (12-bit unit)
    q_conv1d_2_bias                     : 5      (12-bit unit)
    q_conv1d_3_weights                  : 75     (12-bit unit)
    q_conv1d_3_bias                     : 0      (12-bit unit)
    q_dense_weights                     : 3000   (12-bit unit)
    q_dense_bias                        : 6      (12-bit unit)
    q_dense_1_weights                   : 6      (12-bit unit)
    q_dense_1_bias                      : 1      (12-bit unit)
```

Pruning

- Train the trained network and set weights to zero
- If weights are zero, the multiplication is not implemented

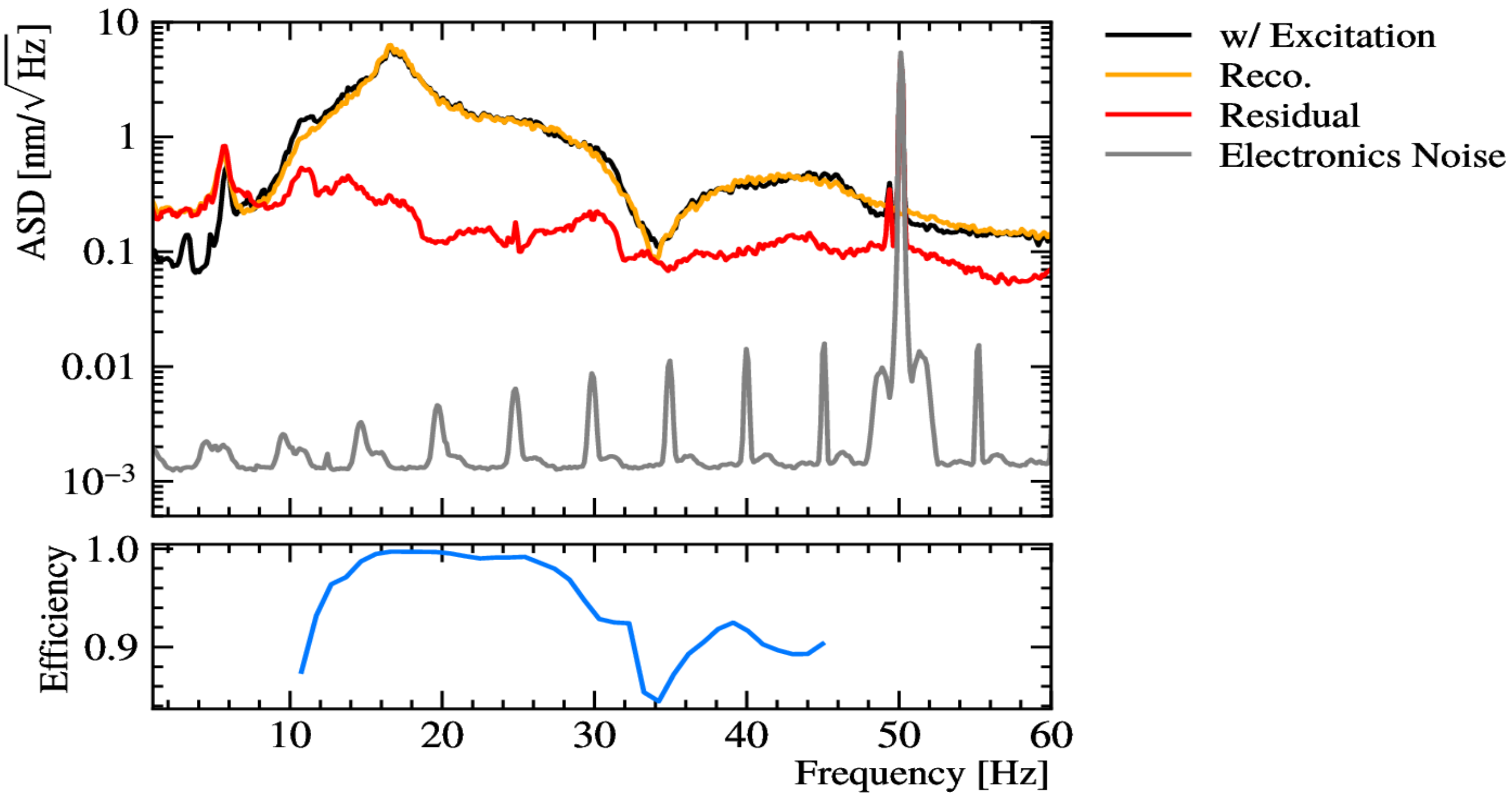
Pruning schedule:

- q_dense: 90%
- q_dense_1: 30%
- others: 80%

Should now fit into the FPGA

```
Weight sparsity:
    q_conv1d                                    : 0.6000
    q_conv1d_1                                  : 0.7867
    q_conv1d_2                                  : 0.7375
    q_conv1d_3                                  : 0.7867
    q_dense                                     : 0.8776
    q_dense_1                                   : 0.2857
    --------------------------------------------
    Total Sparsity                              : 0.8670
```

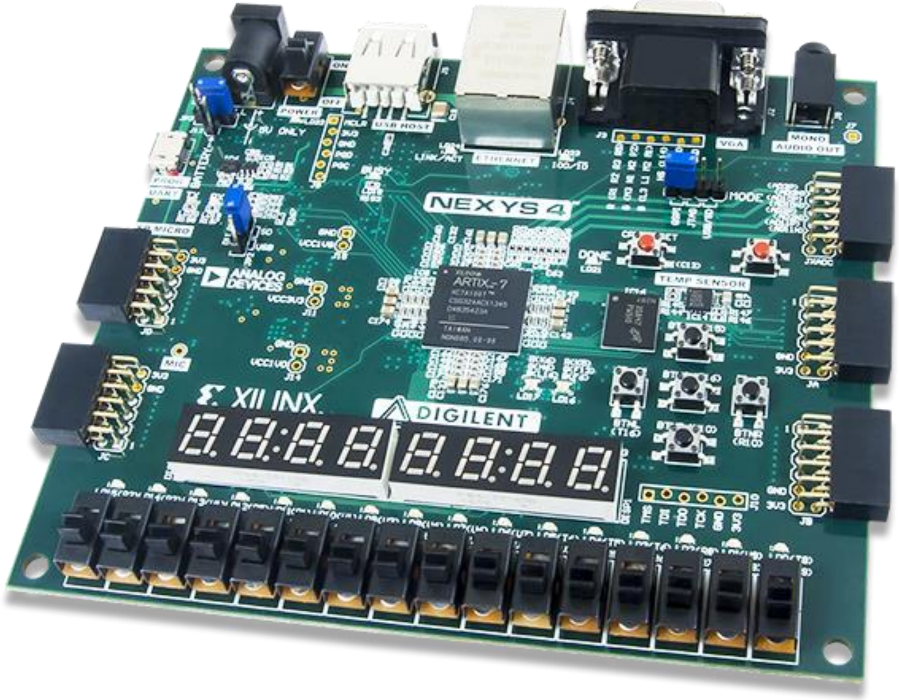
Performance after pruning



Tensorflow and GPU / CPU until here

Implementation in real time

Fixed latency!



hls4ml

GHDL

HDLmake

Vivado™ HLS

manta

ErUM-WAVE

DESY.

BERGISCHE UNIVERSITÄT WUPPERTAL

Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

HZDR
HELMHOLTZ ZENTRUM DRESDEN ROSSENDORF

WWU MÜNSTER

RWTH AACHEN UNIVERSITY

Latency of the ANN itself

All processing needs about 766 clock cycles

- Internal clock: 100 MHz

- **Latency**: time it takes from input to output
- **Interval**: the ANN accepts new data every 715 clock cycles

```
Latency:
  * Summary:
  +---------+---------+----------+----------+-----+-----+----------+
  |  Latency (cycles) |  Latency (absolute) |  Interval | Pipeline |
  |   min   |   max   |    min   |    max   | min | max |   Type   |
  +---------+---------+----------+----------+-----+-----+----------+
  |      765|      766| 7.650 us | 7.660 us |  613|  715| dataflow |
  +---------+---------+----------+----------+-----+-----+----------+
```

Ressource usage

Somehow optimised due to quantisation

- Just for the ANN, the I/O needs also some FF and LUTs

== Utilization Estimates

* Summary:

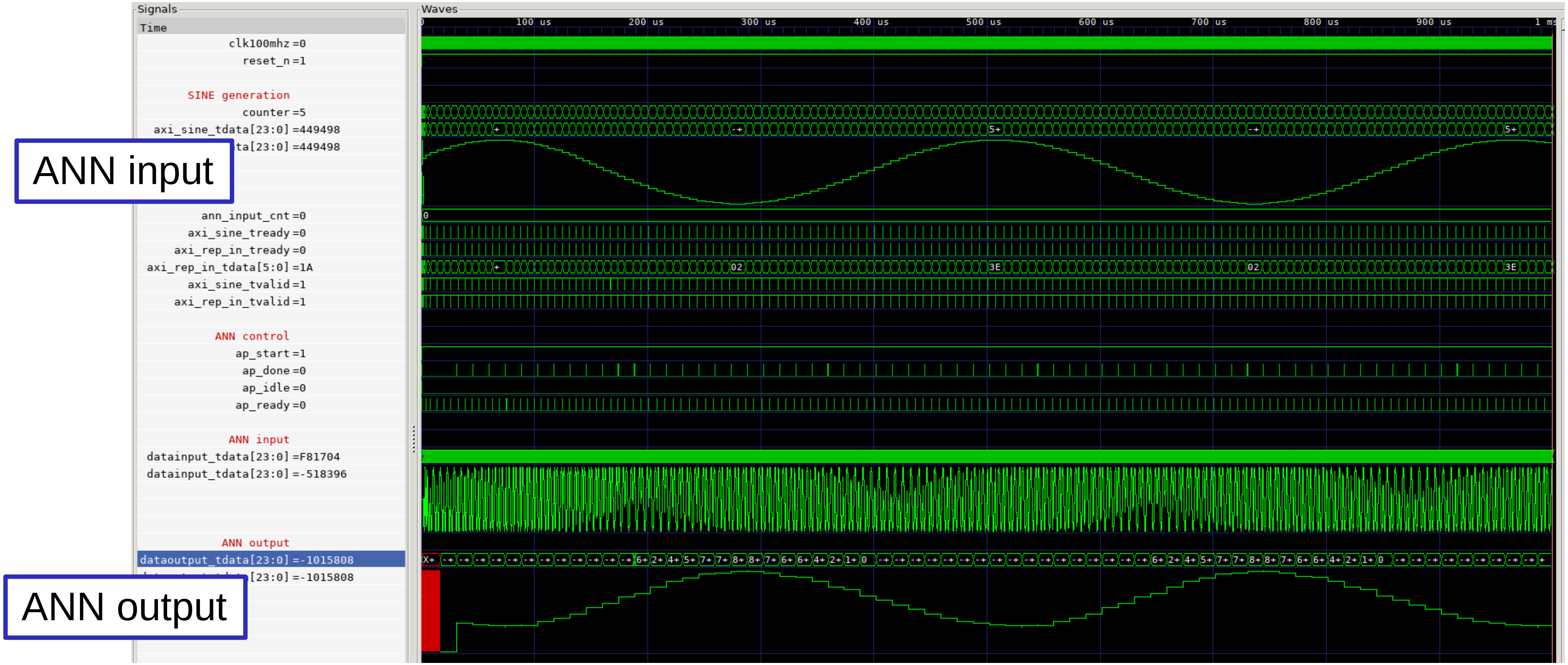
Name	BRAM_18K	DSP48E	FF	LUT	URAM
DSP	-	-	-	-	-
Expression	-	-	0	2	-
FIFO	77	-	2704	3779	-
Instance	50	196	33111	35315	-
Memory	-	-	-	-	-
Multiplexer	-	-	-	-	-
Register	-	-	-	-	-
Total	127	196	35815	39096	0
Available	270	240	126800	63400	0
Utilization (%)	47	81	28	61	0

Utilization Post-Synthesis | **Post-Implementation**

Graph | **Table**

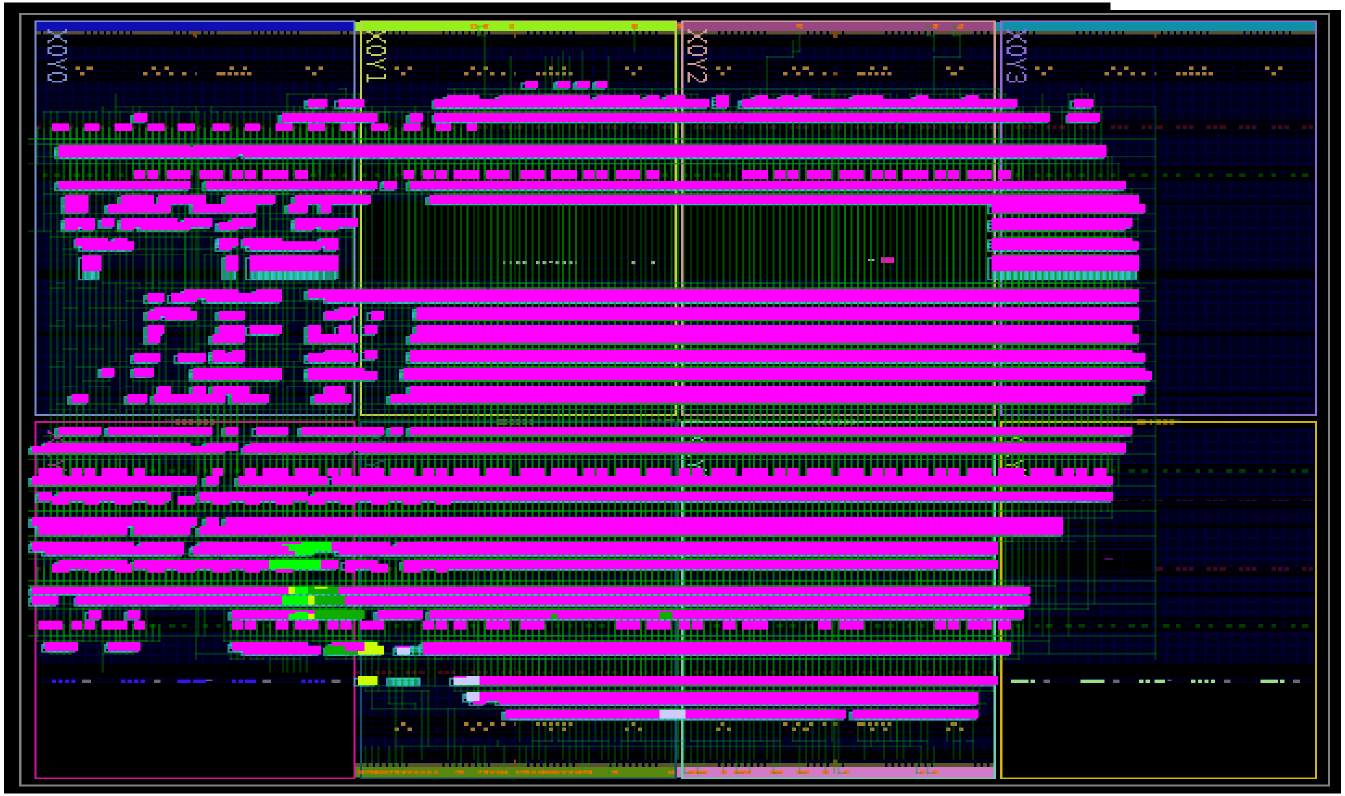
Resource	Utilization	Available	Utilization %
LUT	16639	63400	26.24
LUTRAM	65	19000	0.34
FF	31585	126800	24.91
BRAM	37	135	27.41
DSP	146	240	60.83
IO	39	210	18.57
BUFG	1	32	3.13

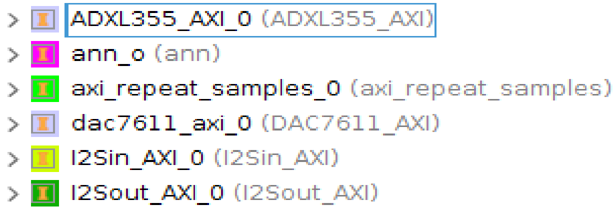
Simulation


Signals
Waves
Time
clk100mhz =0
reset_n =1
SINE generation
counter =5
axi_sine_tdata[23:0] =449498
ta[23:0] =449498
ANN input
ann_input_cnt =0
axi_sine_tready =0
axi_rep_in_tready =0
axi_rep_in_tdata[5:0] =1A
axi_sine_tvalid =1
axi_rep_in_tvalid =1
ANN control
ap_start =1
ap_done =0
ap_idle =0
ap_ready =0
ANN input
datainput_tdata[23:0] =F81704
datainput_tdata[23:0] =-518396
ANN output
dataoutput_tdata[23:0] =-1015808
[23:0] =-1015808
ANN output
100 us
200 us
300 us
400 us
500 us
600 us
700 us
800 us
900 us
1 ms

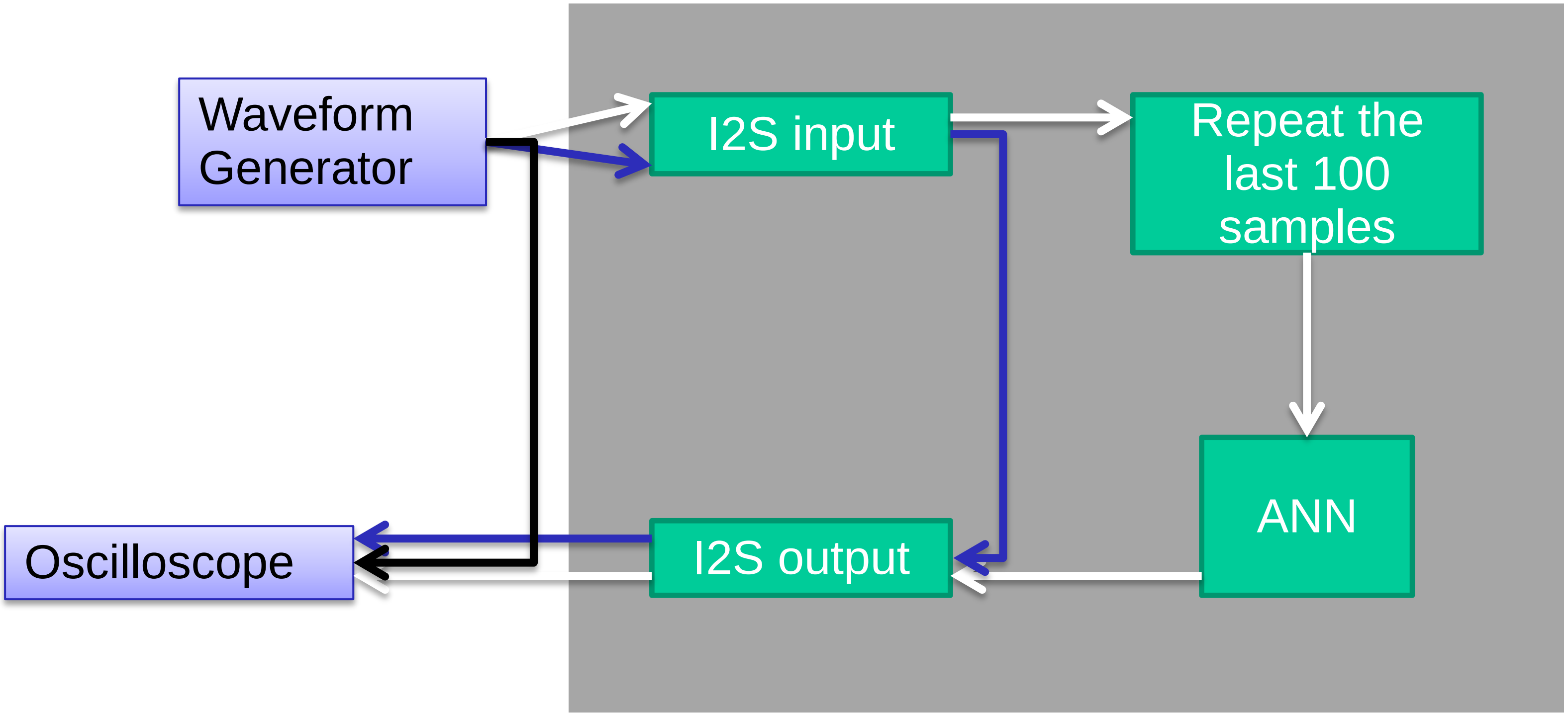
Implementation on the die

Artix 7 A 100

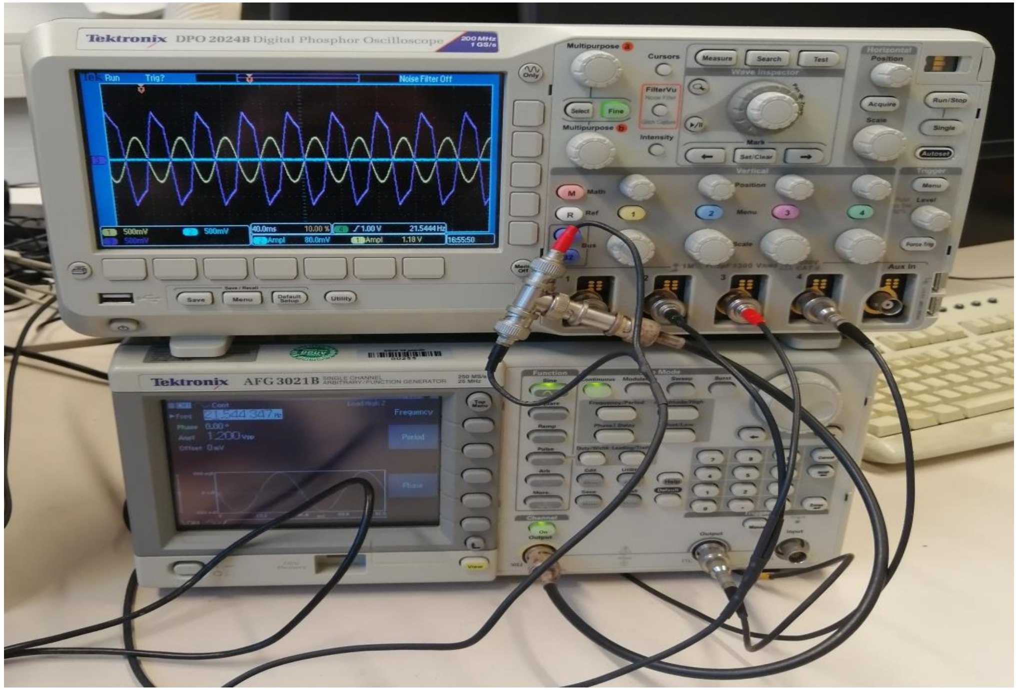




Verification on the FPGA



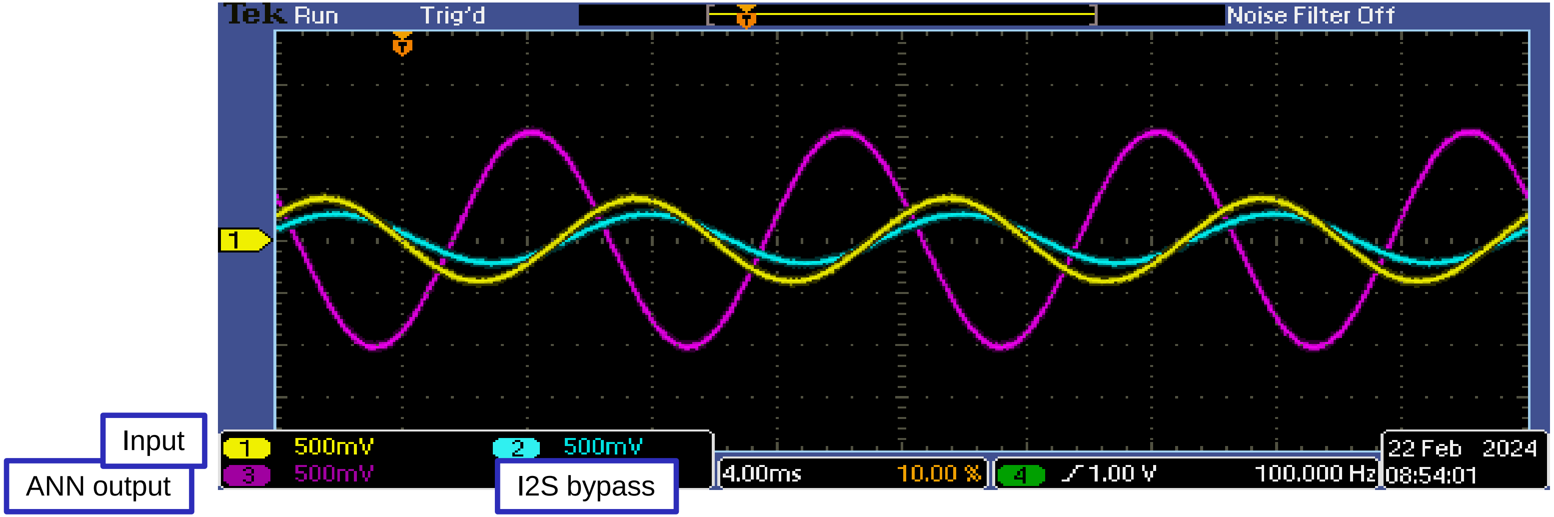
„Lab“ Setup



DESY.
BERGISCHE UNIVERSITÄT WUPPERTAL
Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG
HZDR
HELMHOLTZ ZENTRUM DRESDEN ROSSENDORF
WWU MÜNSTER
RWTH AACHEN UNIVERSITY

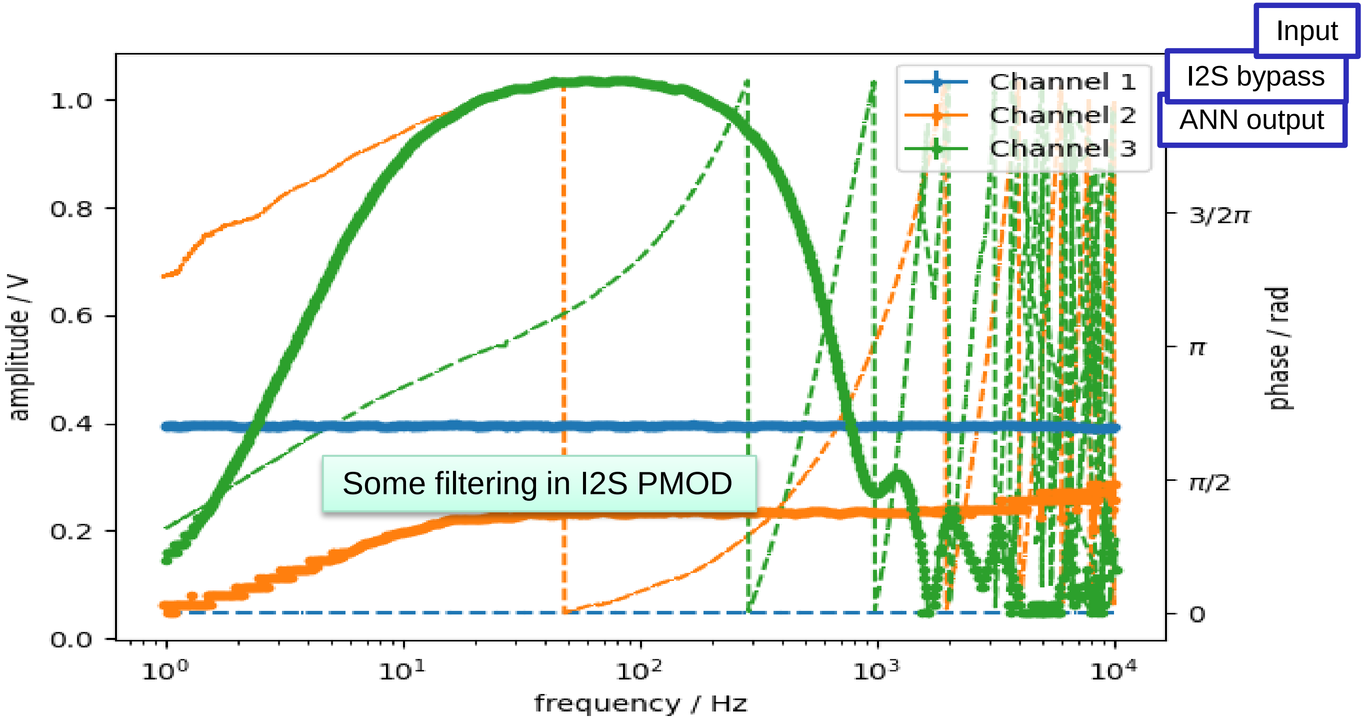
It‘s doing something @ 100 Hz

Samplefrequency is not the designed one



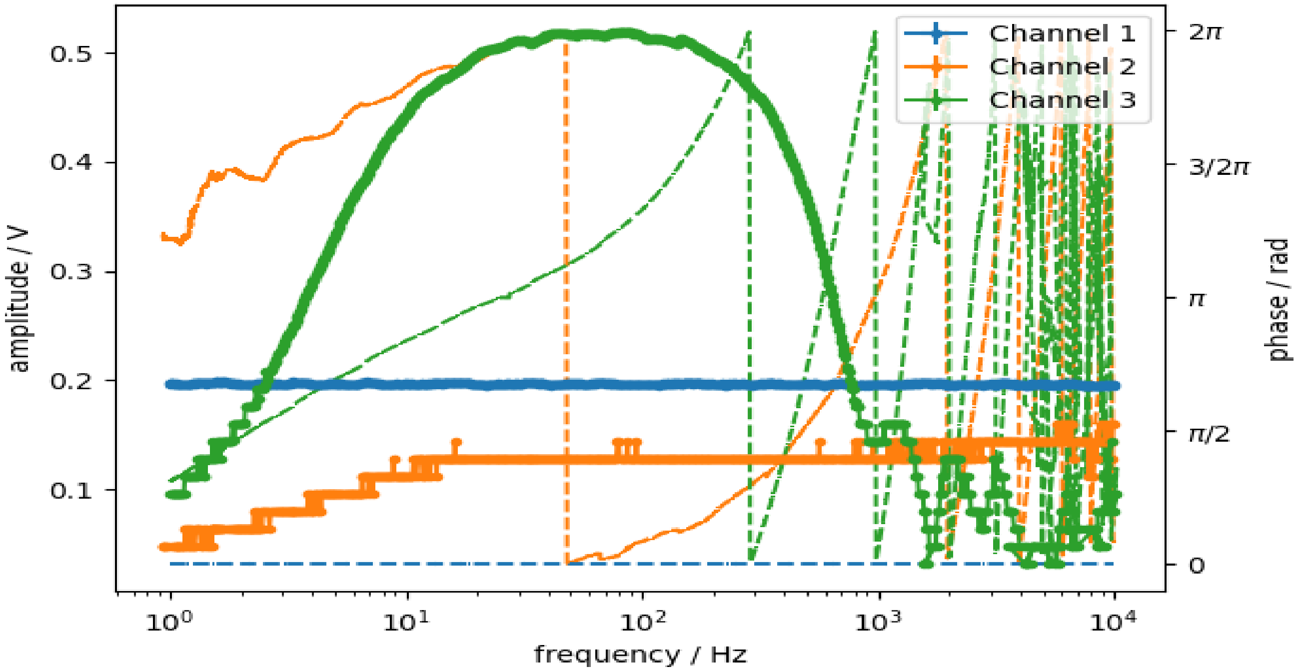
„Frequency“ response

Sine input 0.8Vpp



Similar behaviour for lower amplitude

Input 0.4 Vpp



Summary Active Noise Mitigation

- We have started our toolchain to get the Network onto the FPGA
- Using a sine generator, the network shows some performance
- Next steps are connecting the accelerometers and tuning the output to match the excitation

Thank you!