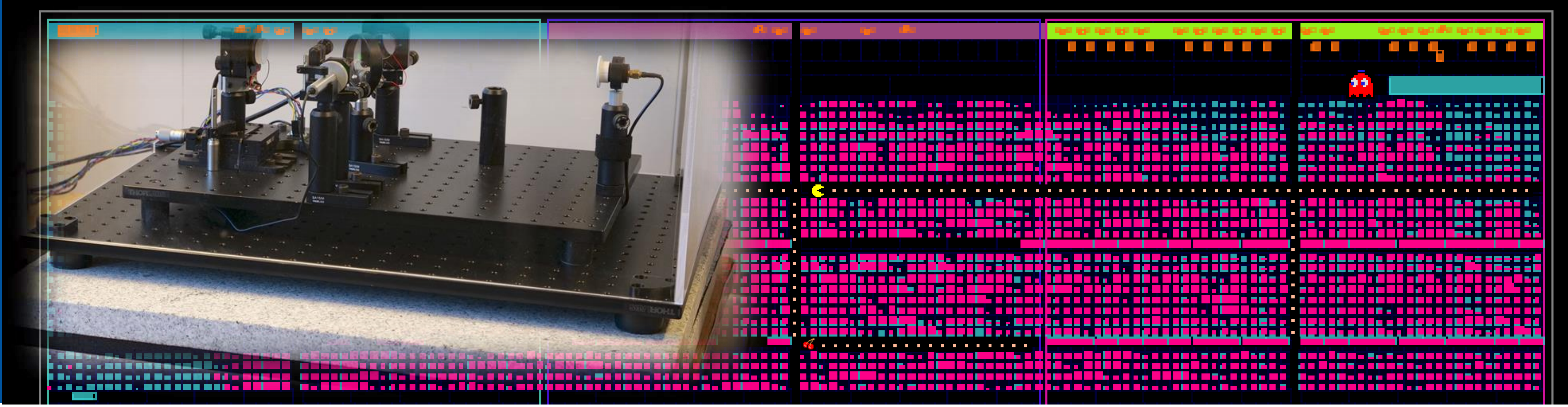


Active Noise Mitigation

… in (almost) realtime …

Markus Bachlechner –Tim Kuhlbusch
Jochen Steinmann – Achim Stahl

RWTH Aachen University

ErUM-Wave General Meeting

27th February 2024

III. Physikalisches Institut B

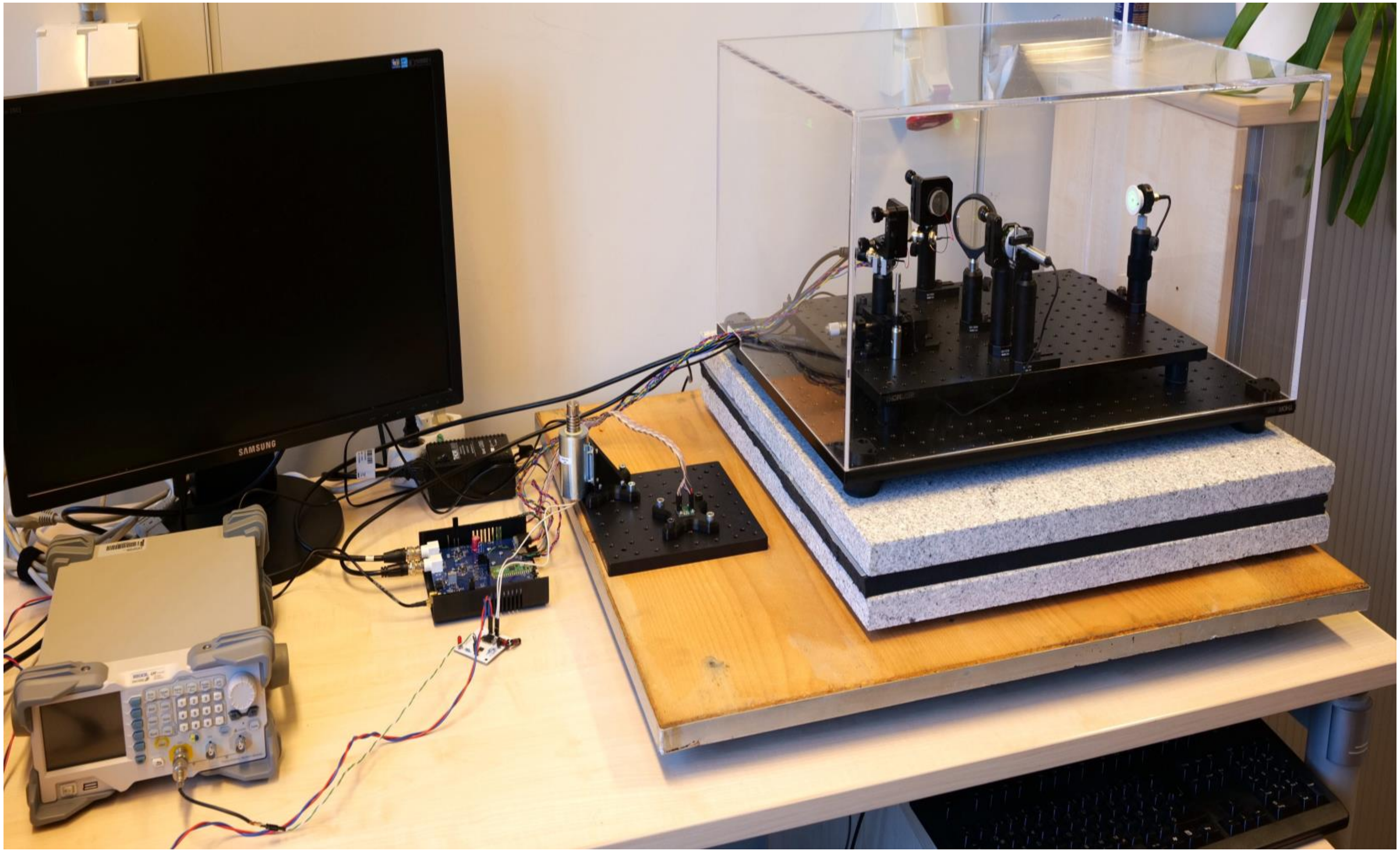
Sandbox

for real time ANN implementations

Gather information in a controlled environment.

Quite an „easy“ task.
Once the toolchain is installed,
more complex networks should be possible as well.

Lab Setup Overview

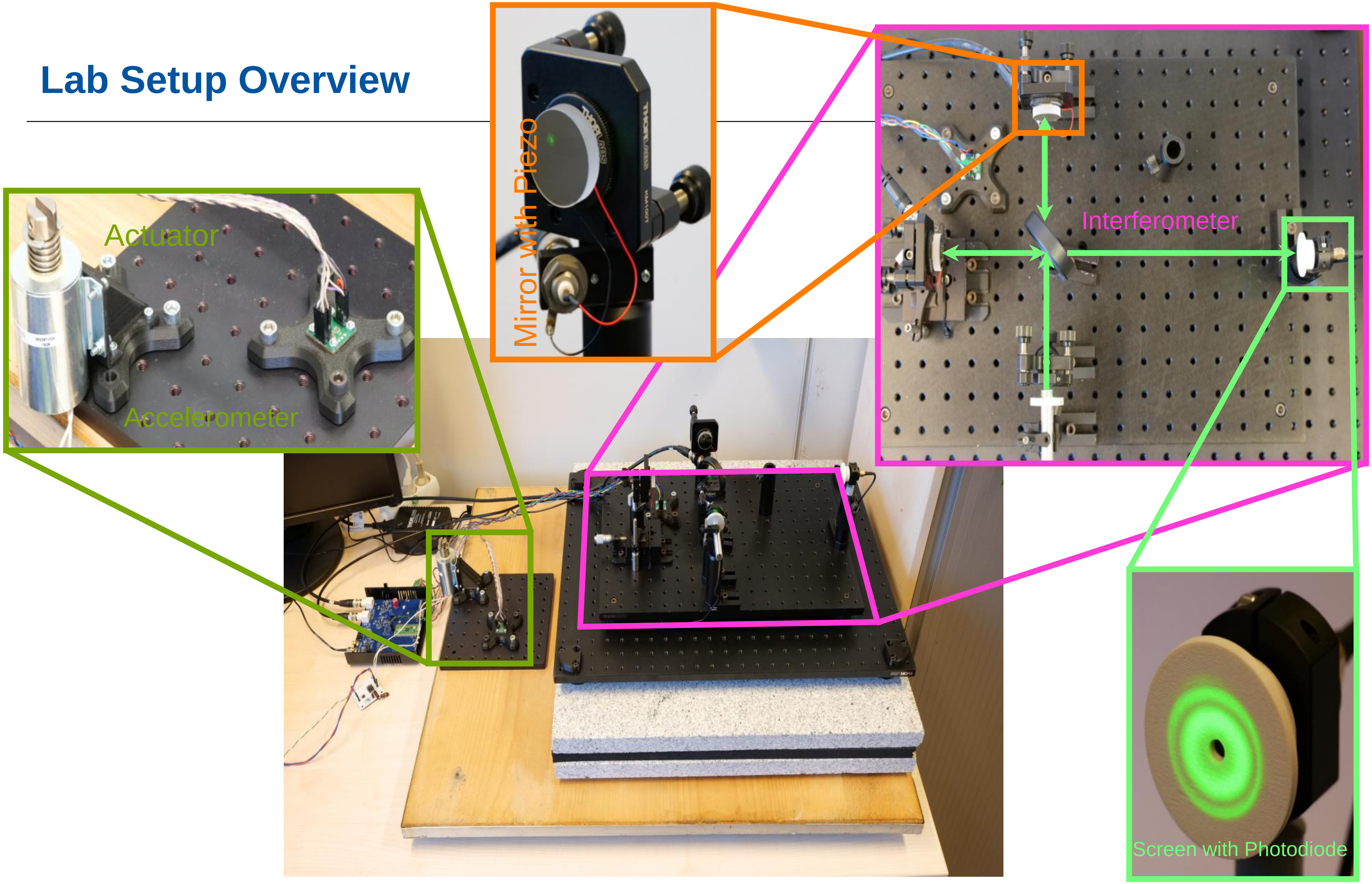


Goal: Online noise mitigation of demonstrator interferometer based on real data

Jochen Steinmann, Markus Bachlechner | RWTH Aachen University

III. Physikalisches Institut B

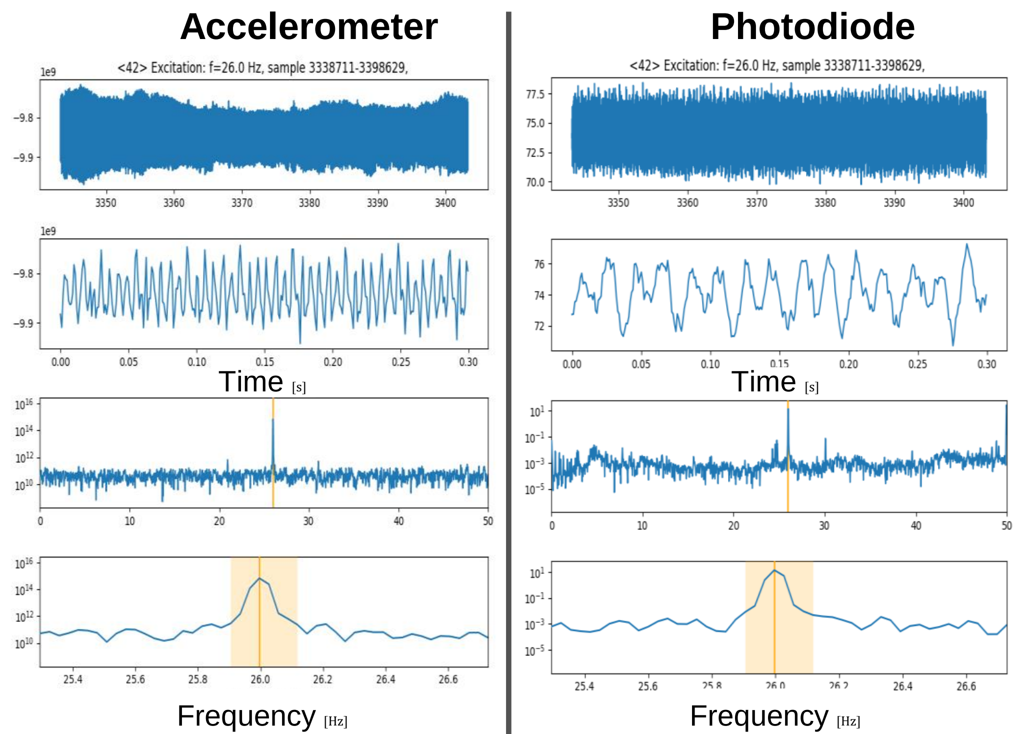
Lab Setup Overview

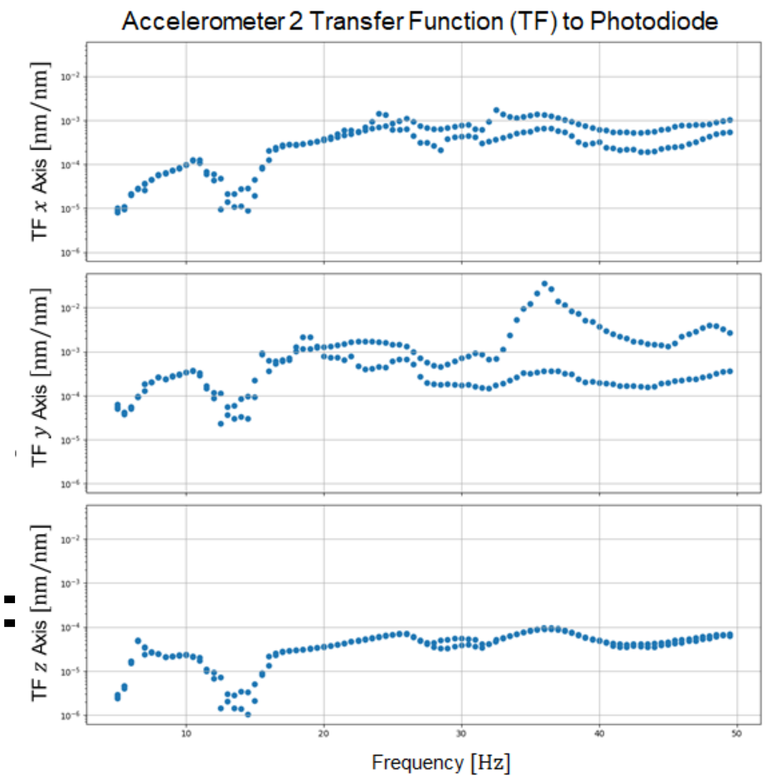


Characterization and Frequency Response

Determination of the Transfer function:

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





III. Physikalisches Institut B
RWTH AACHEN UNIVERSITY

Noise Mitigation Neural Network

III. Physikalisches Institut B

RWTH AACHEN UNIVERSITY

Mitigation Principle and Optimization

Input:

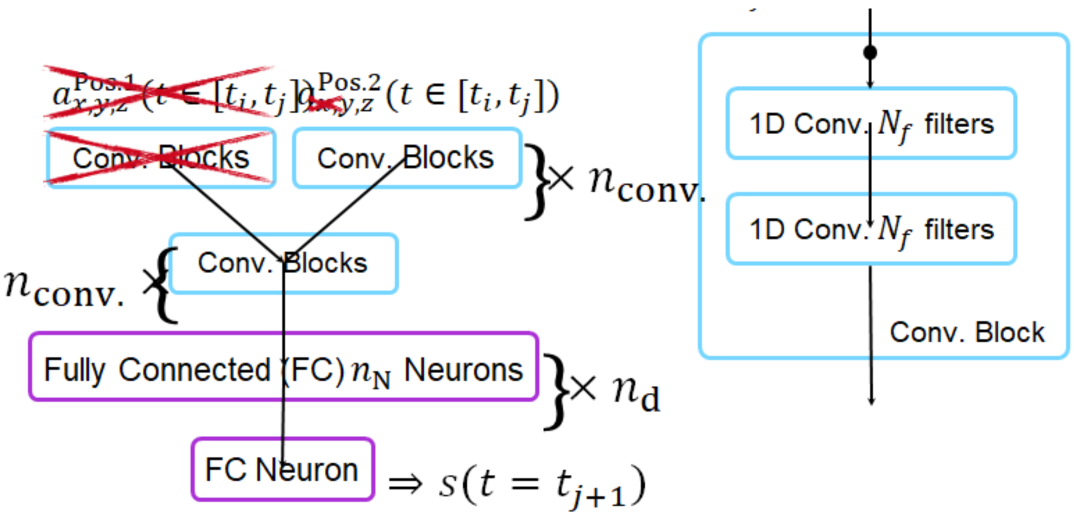
- Measurement accelerometers $a_k^{\mathrm{Pos.1/2}}(t)$ *with* $k \in \{x, y, z\}$
  - Window of duration T ($\sim 100\mathrm{ms}$) from t_i to t_j
  - White noise excited at $10 - 30\mathrm{Hz}$

Output:

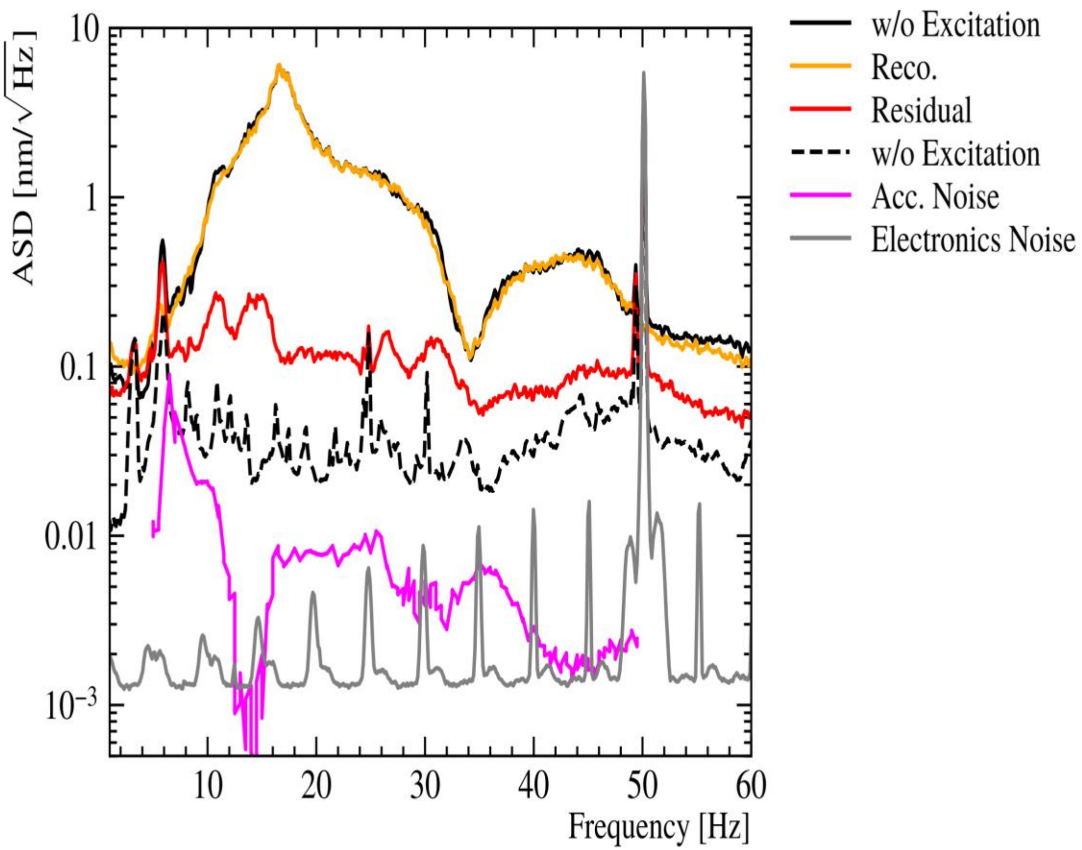
- Prediction for photodiode signal $s(t)$ at time t_{j+1}

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



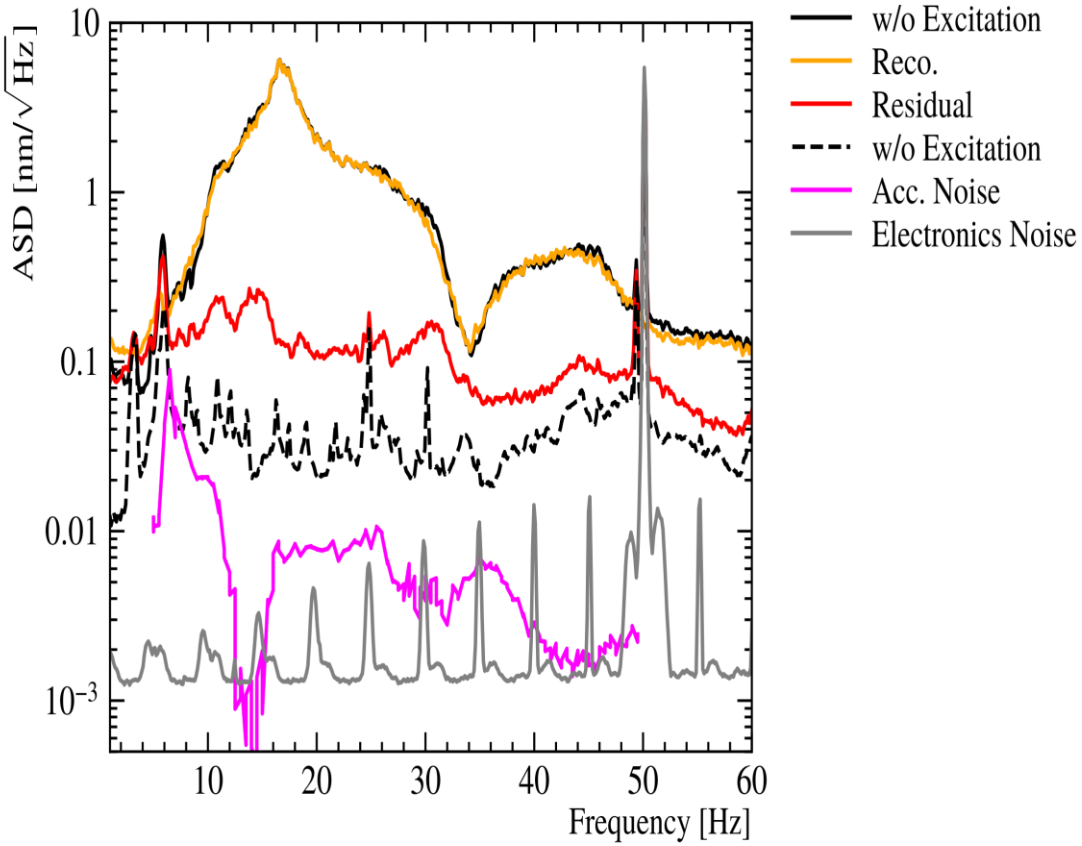
Performance Evaluation - BEFORE Pruning

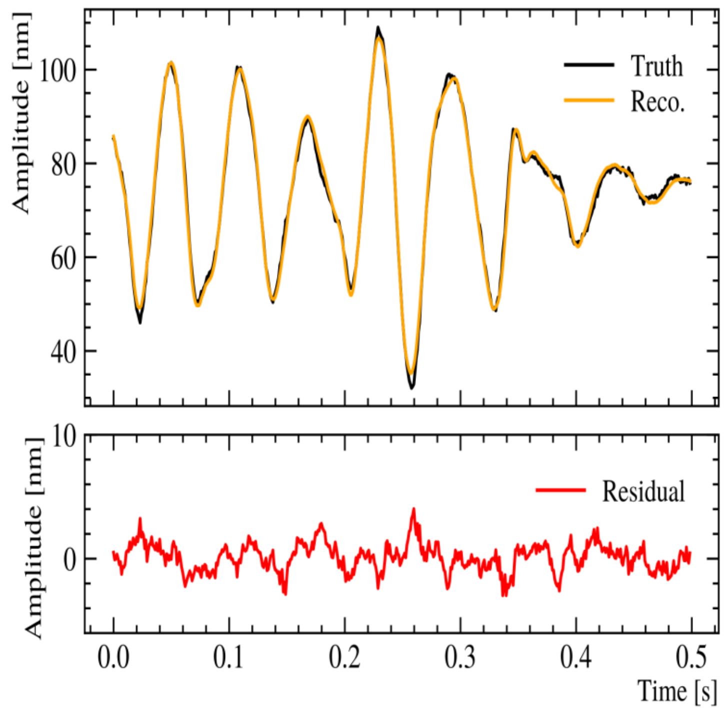


Metric	w/o pruning
$\varepsilon_{\mathrm{tot}}$	95.79 %
$\varepsilon_{\mathrm{w/o\ 50\,Hz}}$	99.15 %

Jochen Steinmann, Markus Bachlechner | RWTH Aachen University

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 %

Jochen Steinmann, Markus Bachlechner | RWTH Aachen University

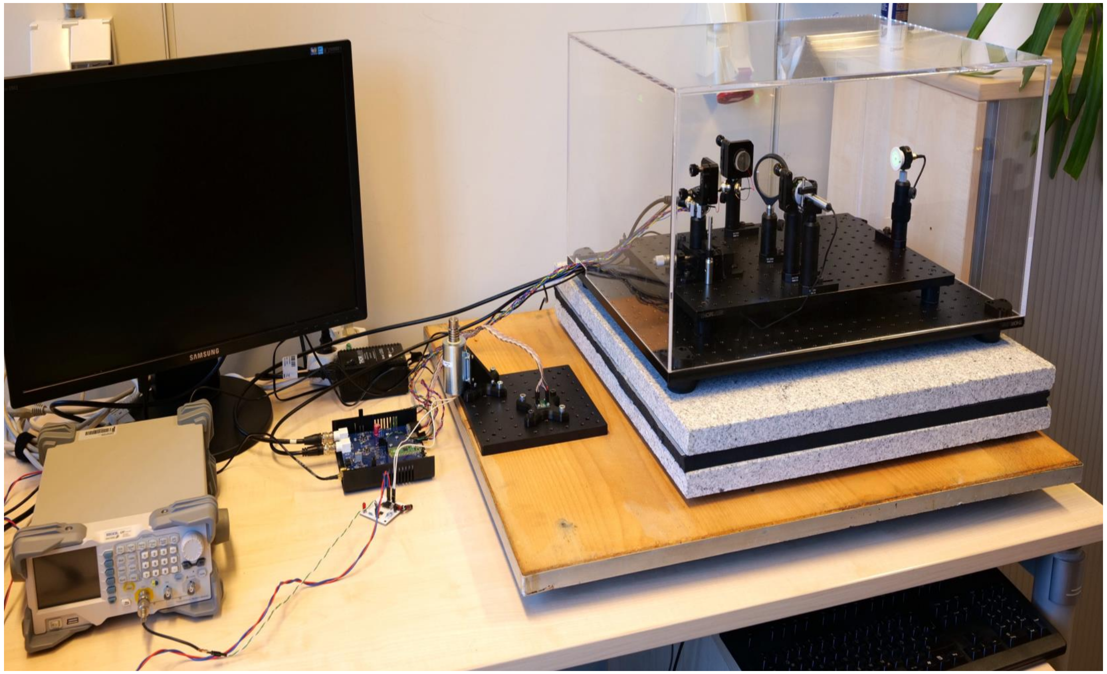
III. Physikalisches Institut B
RWTH AACHEN UNIVERSITY

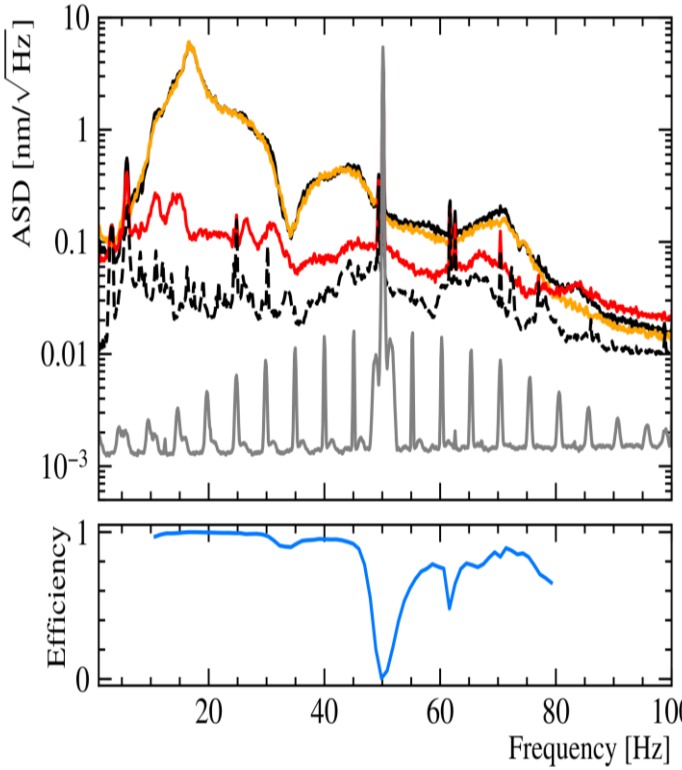
Summary Python Toolchain

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





Jochen Steinmann, Markus Bachlechner | RWTH Aachen University

III. Physikalisches Institut B
RWTH AACHEN UNIVERSITY

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

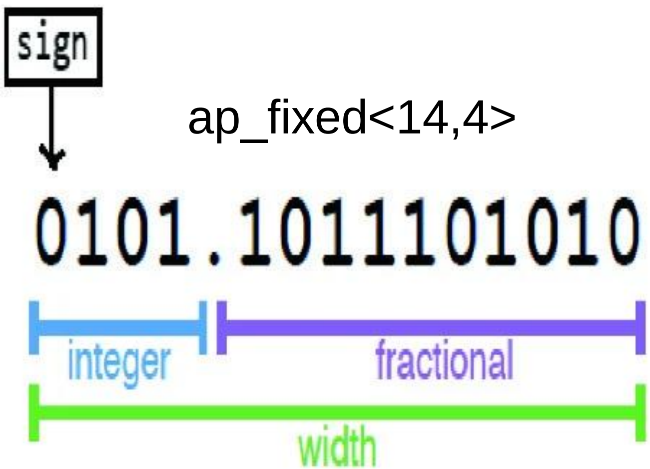
Use the quantisation during training

Quantization

No float on the FPGA

- We have to use either fixed point or integer
  - This can be done layer by layer etc.

ap_fixed<14,4>



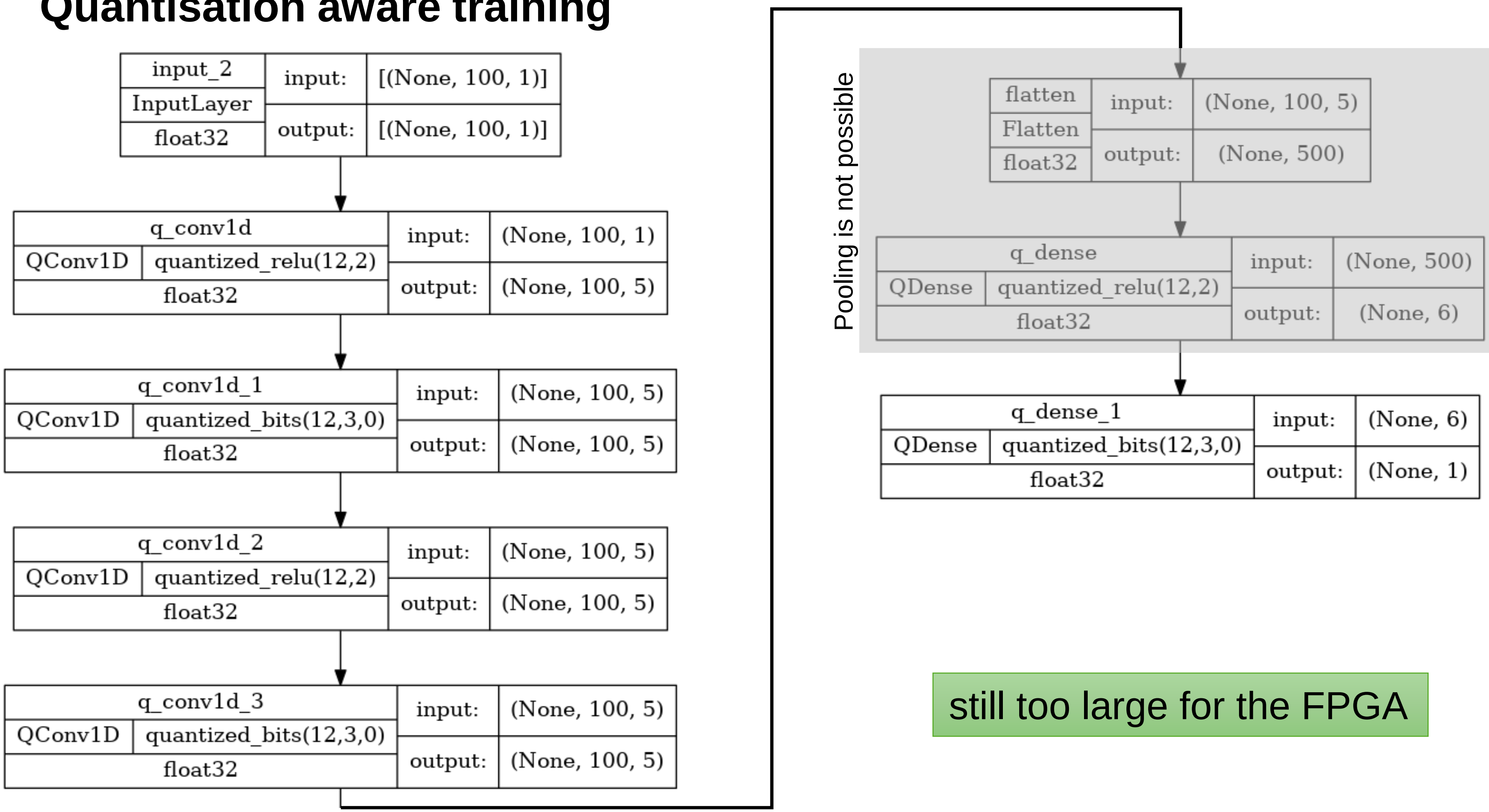
- Two options
  1. Post Training Quantization
  2. **Quantization Aware Training**

Convert the float weights into fixed point during implementation

Use fixed point already during training.

New „reduced“ network

Quantisation aware training



still too large for the FPGA

Jochen Steinmann, Markus Bachlechner | RWTH Aachen University
III. Physikalisches Institut B
RWTH AACHEN UNIVERSITY

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

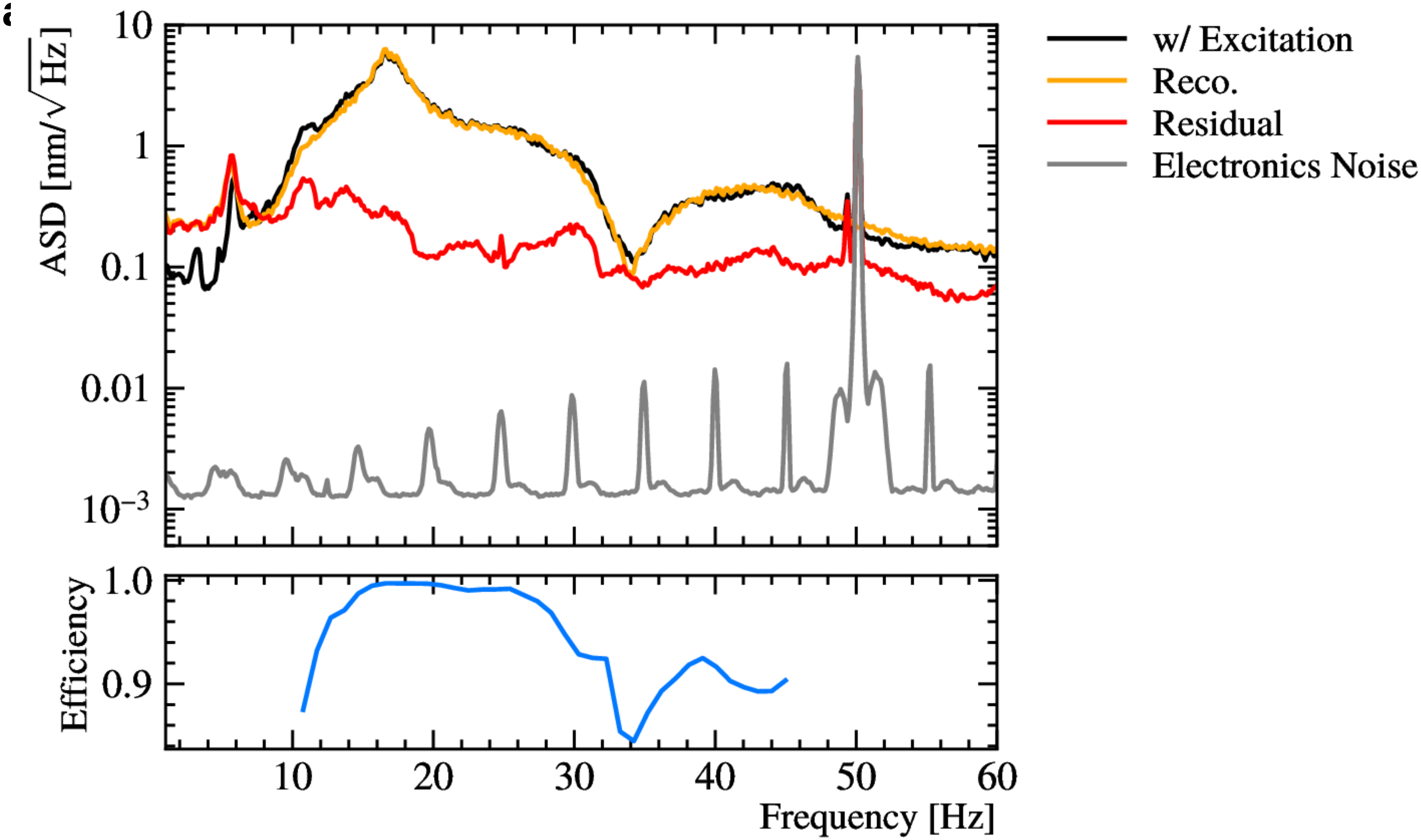
```
Weight sparsity:
    q_conv1d                              : 0.6000
    q_conv1d_1                            : 0.7867
    q_conv1d_2                            : 0.7375
    q_conv1d_3                            : 0.7867
    q_dense                               : 0.8776
    q_dense_1           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

Pruning schedule:

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

Should now fit into the FPGA

Performance after pruning



Jochen Steinmann, Markus Bachlechner | RWTH Aachen University

Tensorflow and GPU / CPU until here

Implementation in (almost) real time

Fixed latency!

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) min	Latency (cycles) max	Latency (absolute) min	Latency (absolute) max	Interval min	Interval max	Pipeline 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:

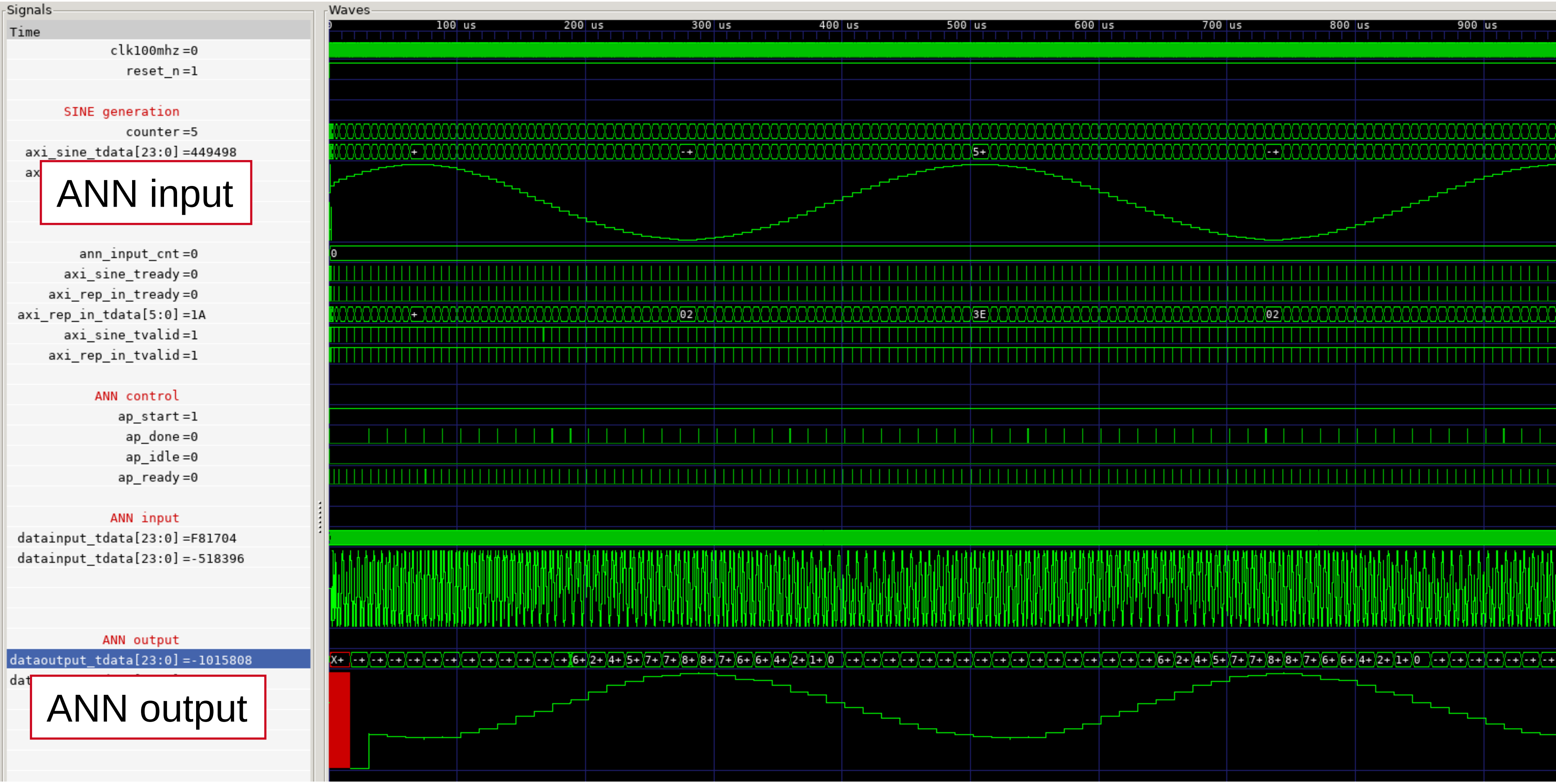
Name	BRAM_18K	DSP48E	FF	LUT	URAM
DSP	-	-	-	-	-
Expression	-	-	0	2	-
FIFO	77	-	2704	3779	-
Instance	50	196	33111	35315	-
Memory	-	-	-	-	-
Multiplexer	-	-	-	-	-
Register	-	-	-	-	-
Total	127	196	35815	39096	
Available	270	240	126800	63400	
Utilization (%)	47	81	28	61	

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

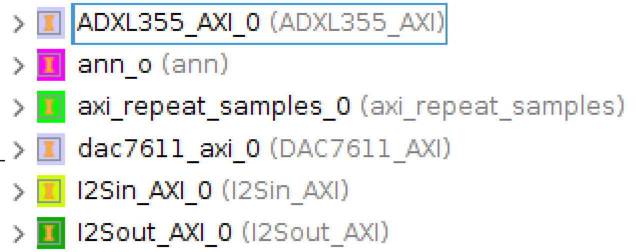
Graph | **Table**

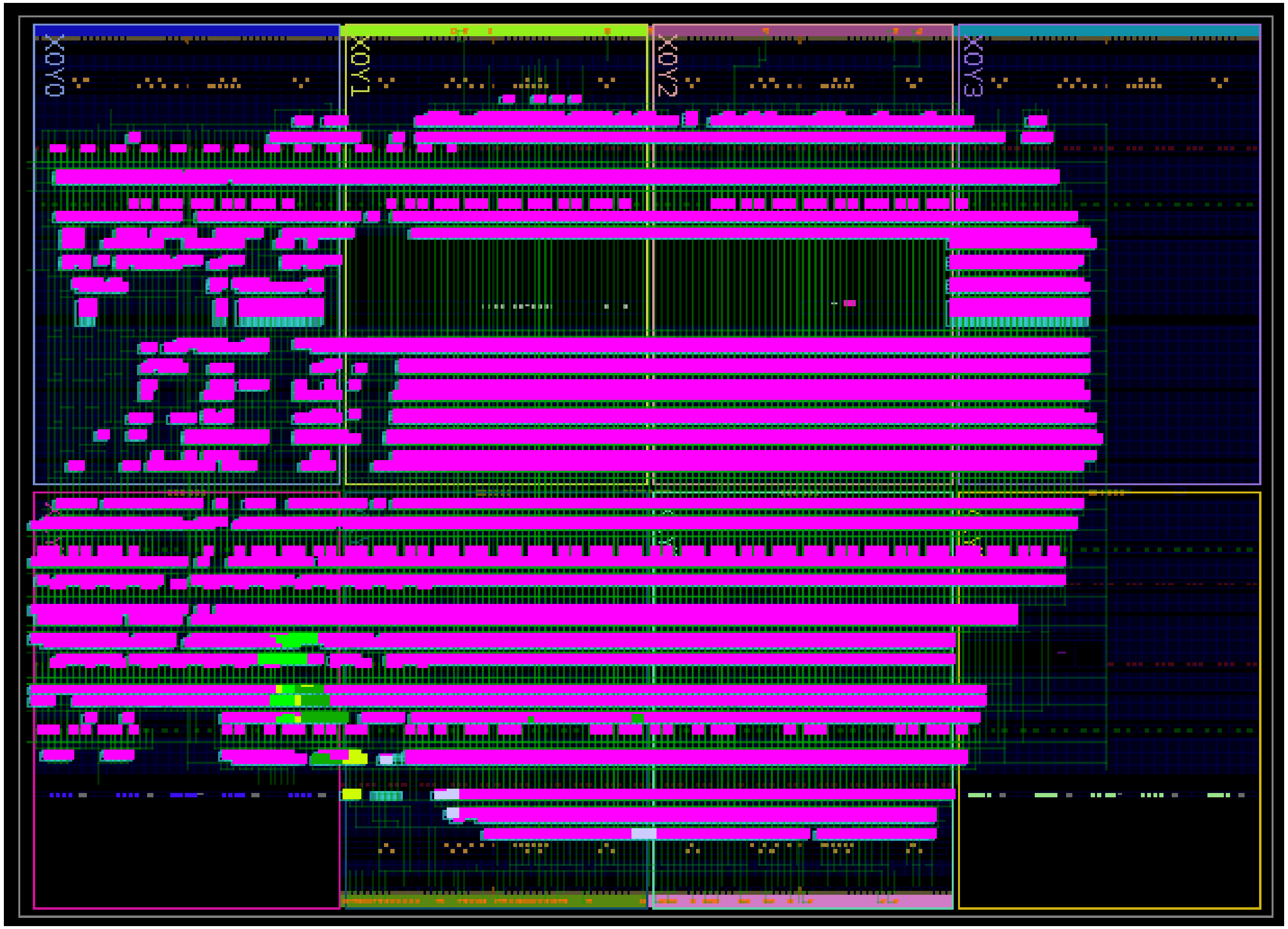
Resource	Utilization	Available	Utilization %
LUT	16639	63400	26.24
LUTRAM	65	19000	0.34
FF	31585	126800	24.91
BRAM	37	135	27.41
DSP	146	240	60.83
IO	39	210	18.57
BUFG	1	32	3.13

Simulation

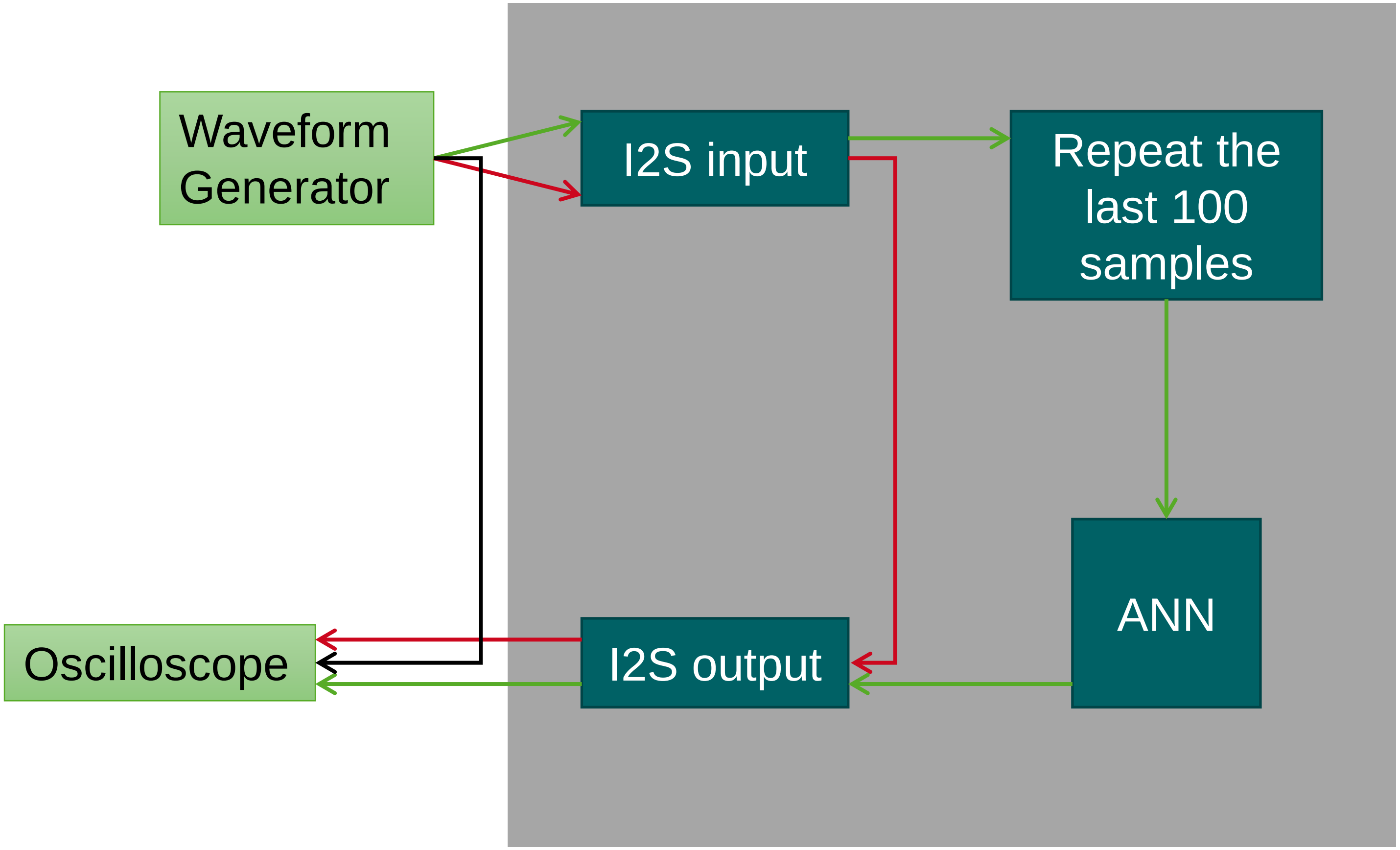

Signals
Time
clk100mhz =0
reset_n =1
SINE generation
counter =5
axi_sine_tdata[23:0] =449498
ANN input
ann_input_cnt =0
axi_sine_tready =0
axi_rep_in_tready =0
axi_rep_in_tdata[5:0] =1A
axi_sine_tvalid =1
axi_rep_in_tvalid =1
ANN control
ap_start =1
ap_done =0
ap_idle =0
ap_ready =0
ANN input
datainput_tdata[23:0]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

Implementation on the die



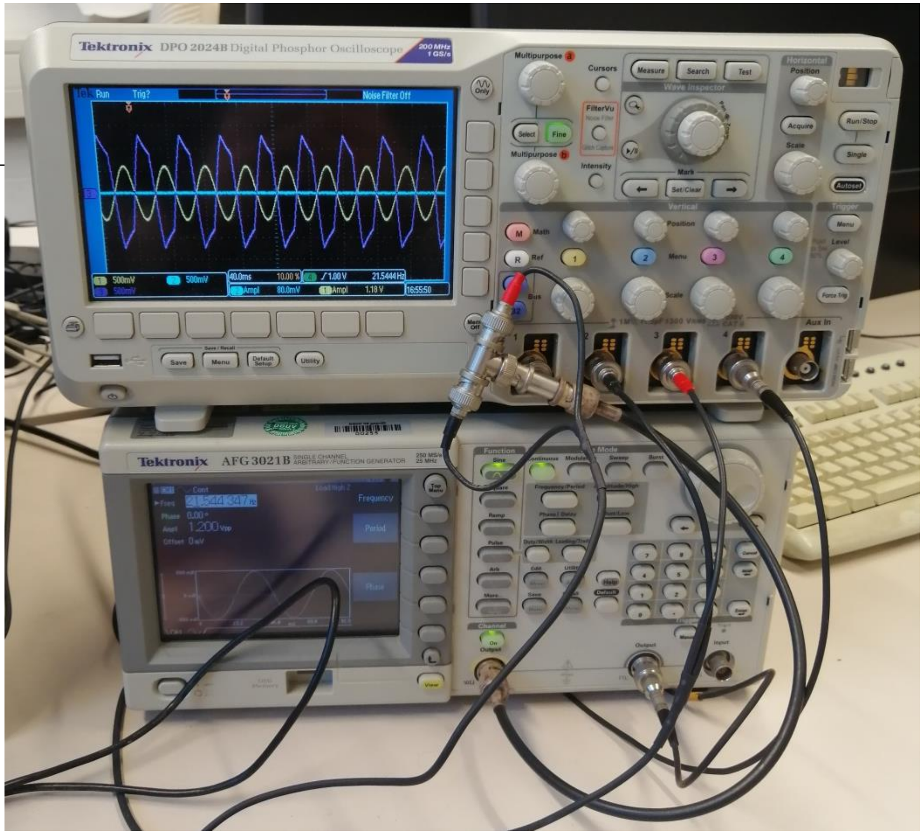


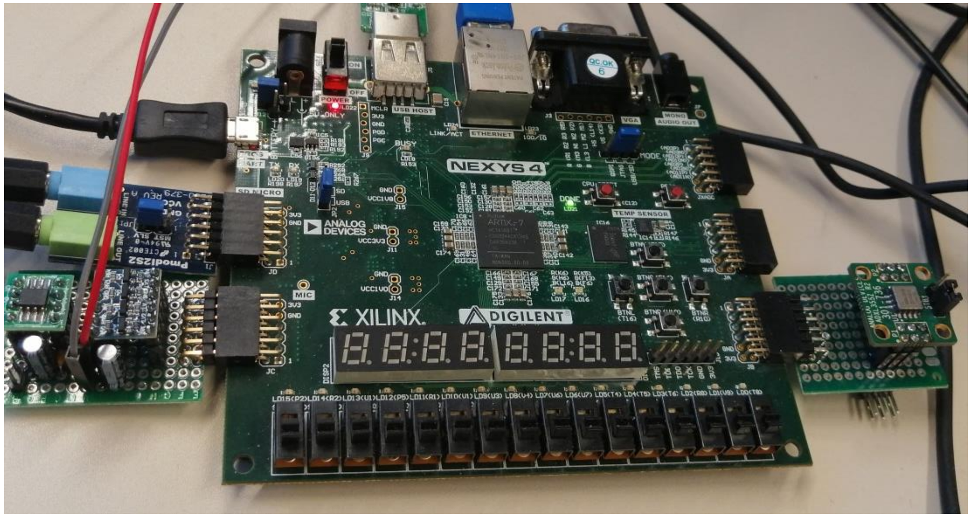
Verification on the FPGA



III. Physikalisches Institut B
RWTH AACHEN UNIVERSITY

„Lab“ Setup

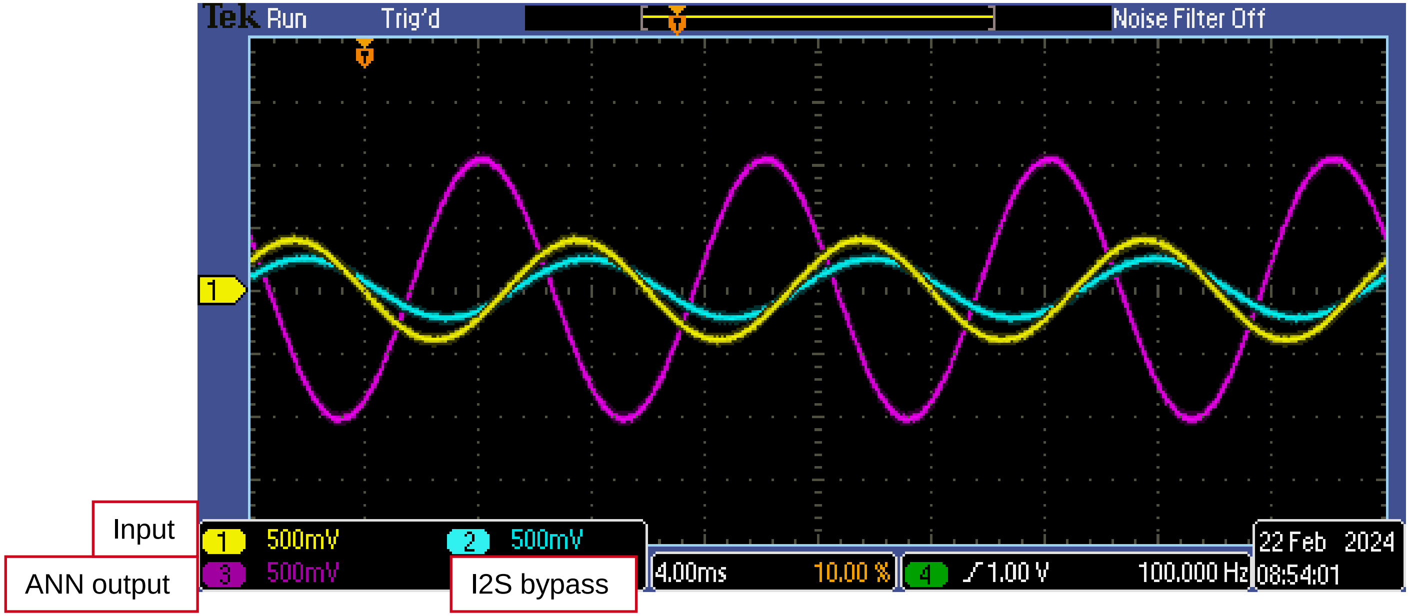




Jochen Steinmann, Markus Bachlechner | RWTH Aachen University

It‘s doing something @ 100 Hz

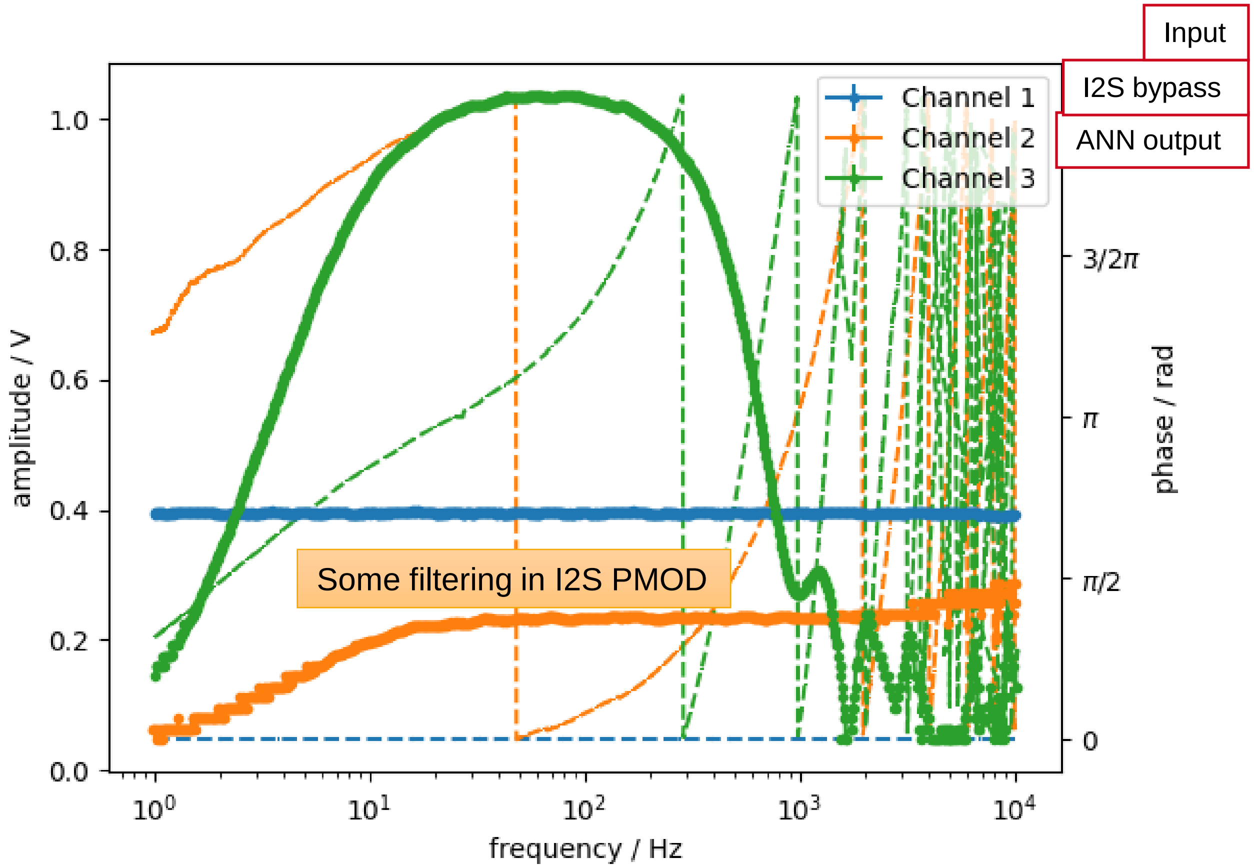
Samplefrequency is not the designed one



Jochen Steinmann, Markus Bachlechner | RWTH Aachen University

„Frequency“ response

Sine input 0.8Vpp

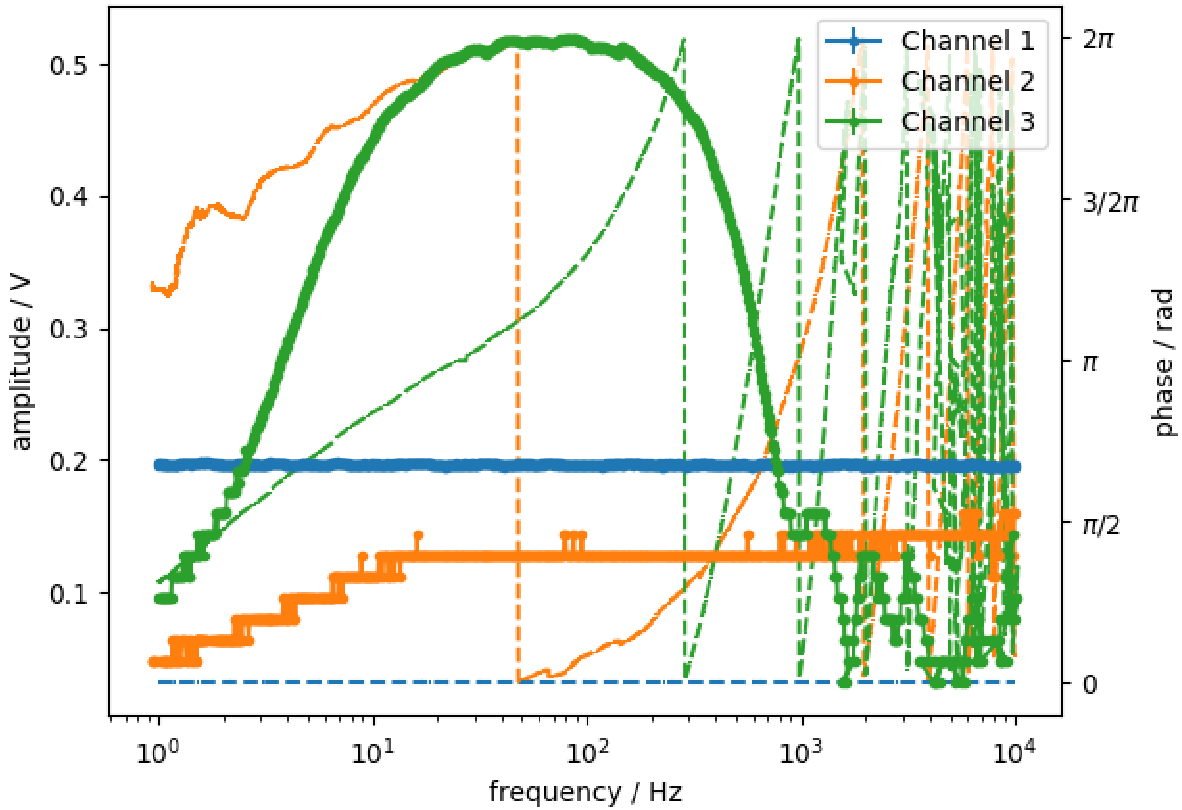


Fitted sine to all channels – no FFT

Jochen Steinmann, Markus Bachlechner | RWTH Aachen University
III. Physikalisches Institut B
RWTH AACHEN UNIVERSITY

Similar behaviour for lower amplitude

Input 0.4 Vpp



Jochen Steinmann, Markus Bachlechner | RWTH Aachen University

Thank you!