

Measuring noise below the instrument noise floor

Designing a dual subtractive cross-correlating
phase noise analyzer

Who am I and why this project



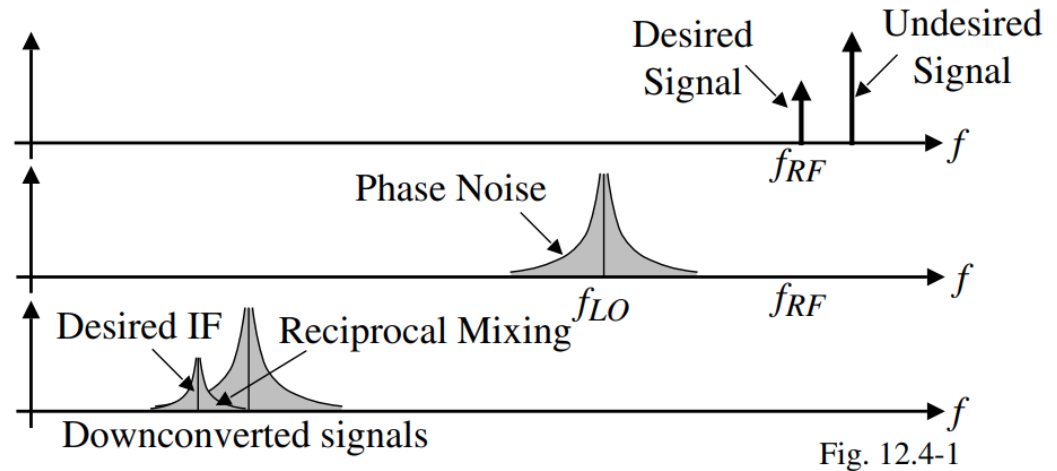
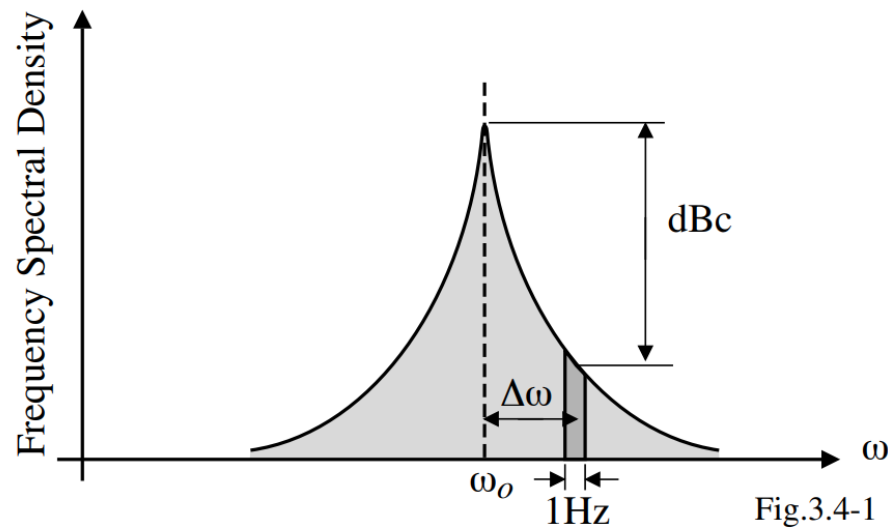
- Erik, PDOEK
 - Started 6 years ago with designing measurement equipment as a new hobby
 - First for own use, later, because people wanted to buy what I built, as products.
- I wanted to:
 - Be able to measure phase noise, including from my best oscillators
 - Home build is much more fun than buying one.
 - Do something with fast ADC and FPGA

Content

- What is phase noise
- The measurement problem
- Theoretical solutions
- Translation to signal processing
- Mapping to hardware
- Hardware limitations
- Implementation approach
- Testing
- “Free” functionality
- Examples
- Next steps
- Conclusion

What is phase noise

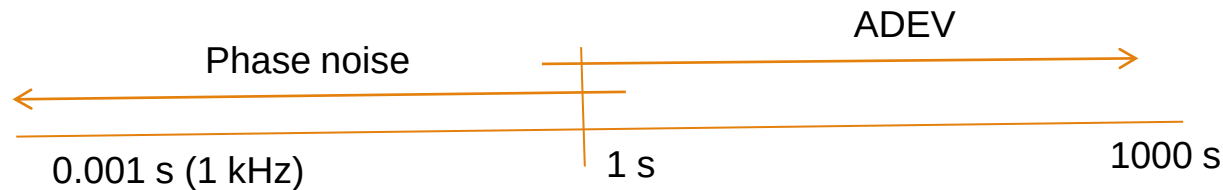
- Every oscillator has some level of frequency instability which leads to spectral “blurring”
- This blurring causes problems in many applications such as receivers
- It is measured as the spectral power in dBc in a 1 Hz BW at a certain offset frequency



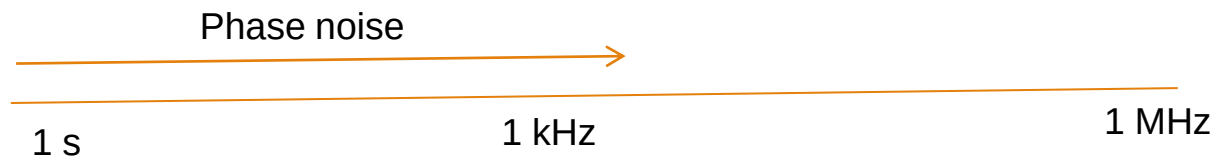
ADEV, Phase noise, what is the difference?



- Timescale determines how oscillator stability is assessed
- Time scale plotted in log seconds gives:



- Phase noise plots typically use log frequency as X-axis so the X-axis is swapped versus above.

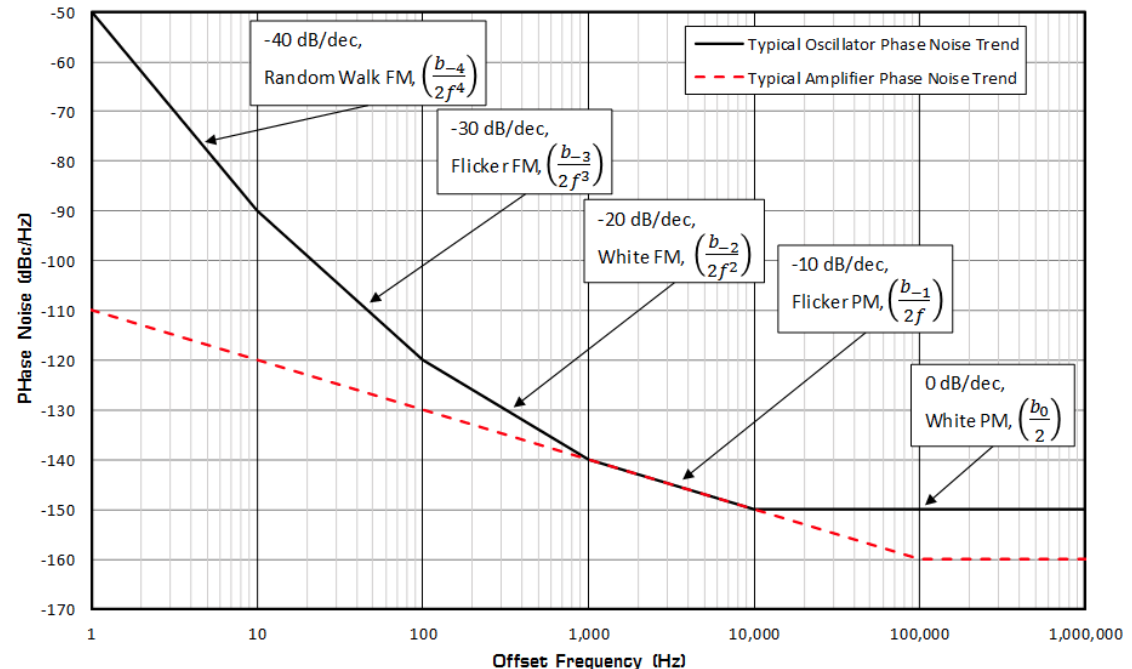


Presenting phase noise

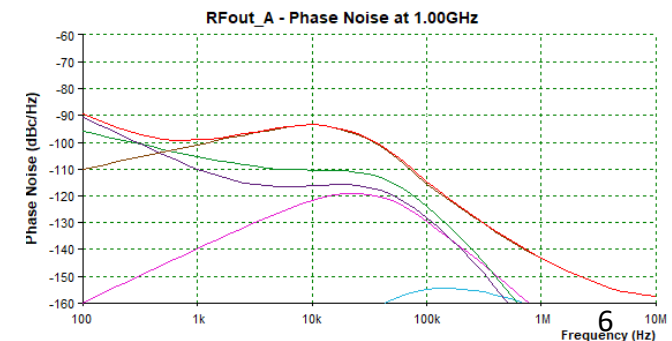


- Plots power versus frequency offset
- Uses log-log plot
 - dB is log power
 - Slope identifies noise source
- Shape is defined by physical characteristics of oscillator
- On top of the reference oscillator a PLL adds various other phase noise contributors and has always a worse phase noise inside the loop bandwidth

OCXO

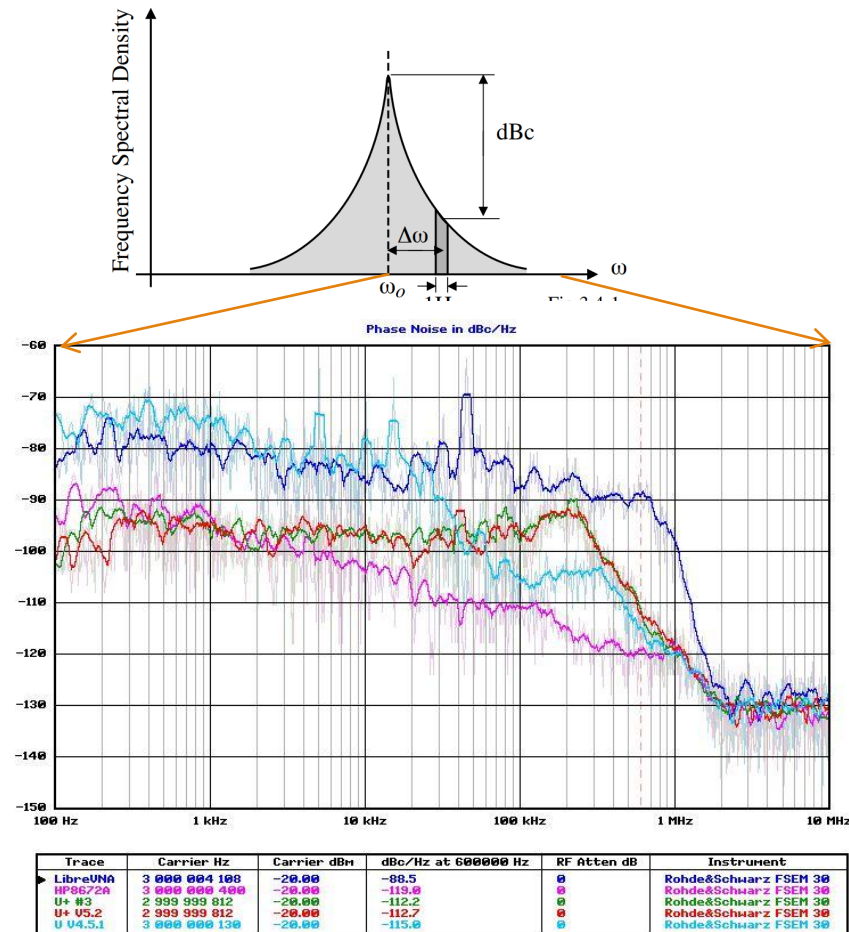


PLL



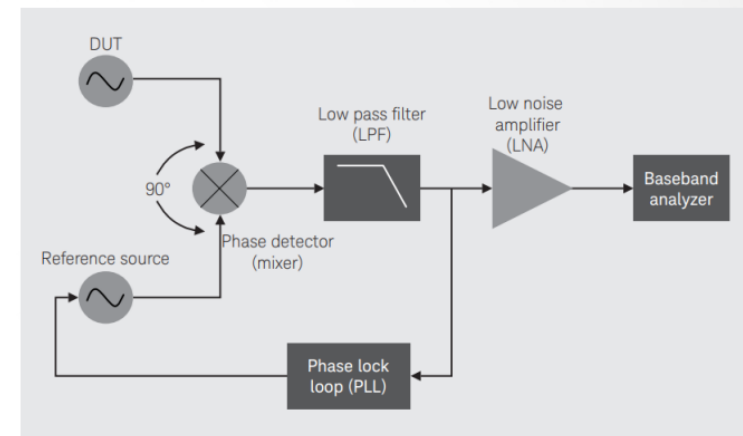
How to measure phase noise(1)

- Use a spectrum analyzer to measure the spectrum of the oscillator output.
- What if the phase noise of the oscillator is (much) smaller than the phase noise of the spectrum analyzer?
 - A typical PLL used to determine the spectrum analyzer input frequency has a much larger phase noise than a good OCXO
- A spectrum analyzer can't measure the true phase noise of a good OCXO



How to measure phase noise(2)

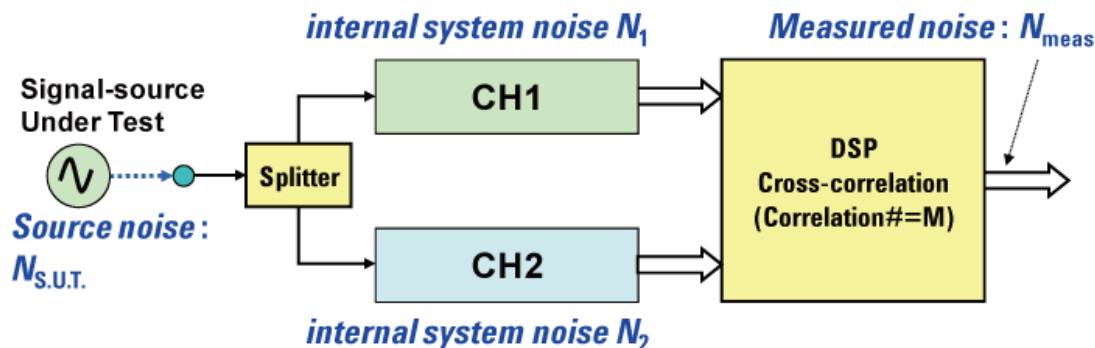
- A more advanced method is to lock a very stable oscillator (REF) to the to be measured oscillator (DUT) and down convert the oscillator output to zero Hz effectively **eliminating the carrier**.
- After the low pass filter and the LNA you can measure the baseband (audio) spectrum without the big carrier peak
- Two problems:
 - REF phase noise must be below DUT phase noise
 - REF frequency must be equal to DUT frequency
- You still can't measure a DUT that is better than the REF



Home-built phase noise analyzer

Eliminating the REF noise

- Split DUT
- Duplicate measurement
- Use two independent REFs
- Cross correlate output
- Only DUT is common
- REF noise reduced
- DUT noise survives



$$N_{meas} = N_{S.U.T.} + (N_1 + N_2) / \sqrt{M} \quad \text{Assuming } N_1 \text{ and } N_2 \text{ are uncorrelated.}$$

M (number of correlation)	10	100	1,000	10,000
Noise reduction on $(N_1 + N_2)$	-5dB	-10dB	-15dB	-20dB

ssa0210

- M averages give SQRT(N) REF noise reduction
- System grows to 3 inputs (DUT, REF1, REF2)
- **With enough measurement time you can measure up to 20 dB below the noise of the reference oscillators**
- Problem: Doing 10000 FFT's can take a long time!

Enabling DUT/REF Frequency difference

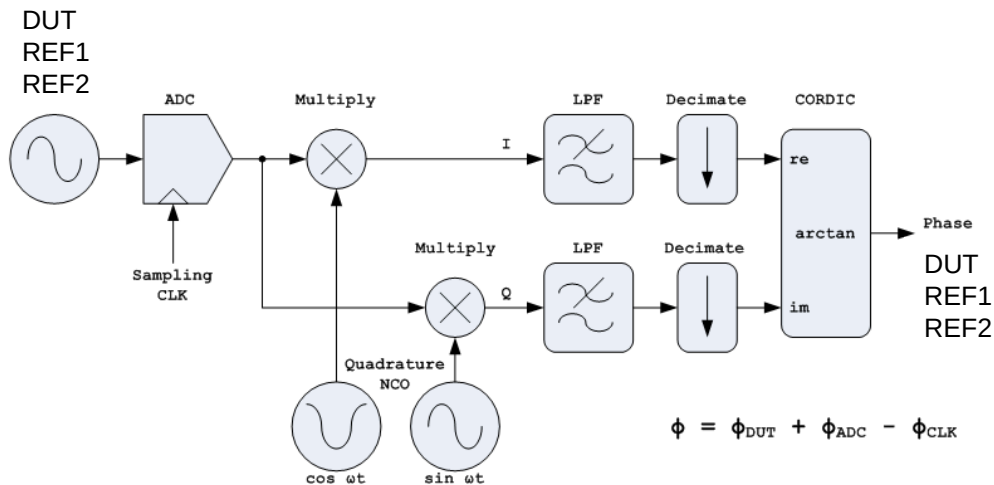


- Don't mix DUT and REF with each other but mix both DUT and REF to zero Hz using separate LO's
 - Scale calculated phase before phase subtraction to compensate for LO frequency difference.
- Phase subtraction still eliminates the noise of the LO's
 - $\text{Noise} = \text{Scaled}(\text{Phase}(\text{DUT})) - \text{Phase}(\text{REF})$
- This only works if the LO's at different frequencies are still perfectly correlated which is impossible in the analog world
- Can we do this with digital signal processing?

Translation to DSP(1)

For each input:

- Digitize using a fast ADC
- Numerically Controlled Oscillators (NCO) locked to ADC sampling clock
- Use I/Q down mix and CORDIC to measure phase against ADC clock of DUT, REF1 and REF2

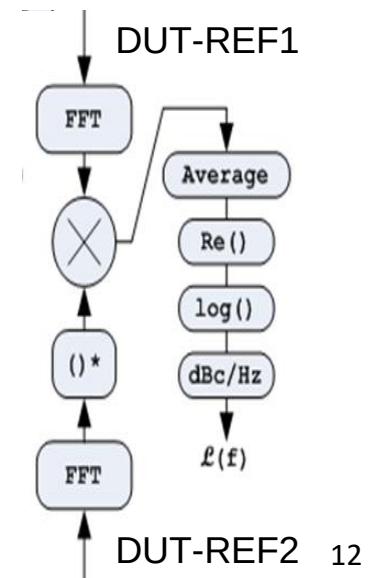


Calculate DUT-REF1 and DUT-REF2

- This removes the ADC clock in each pair.

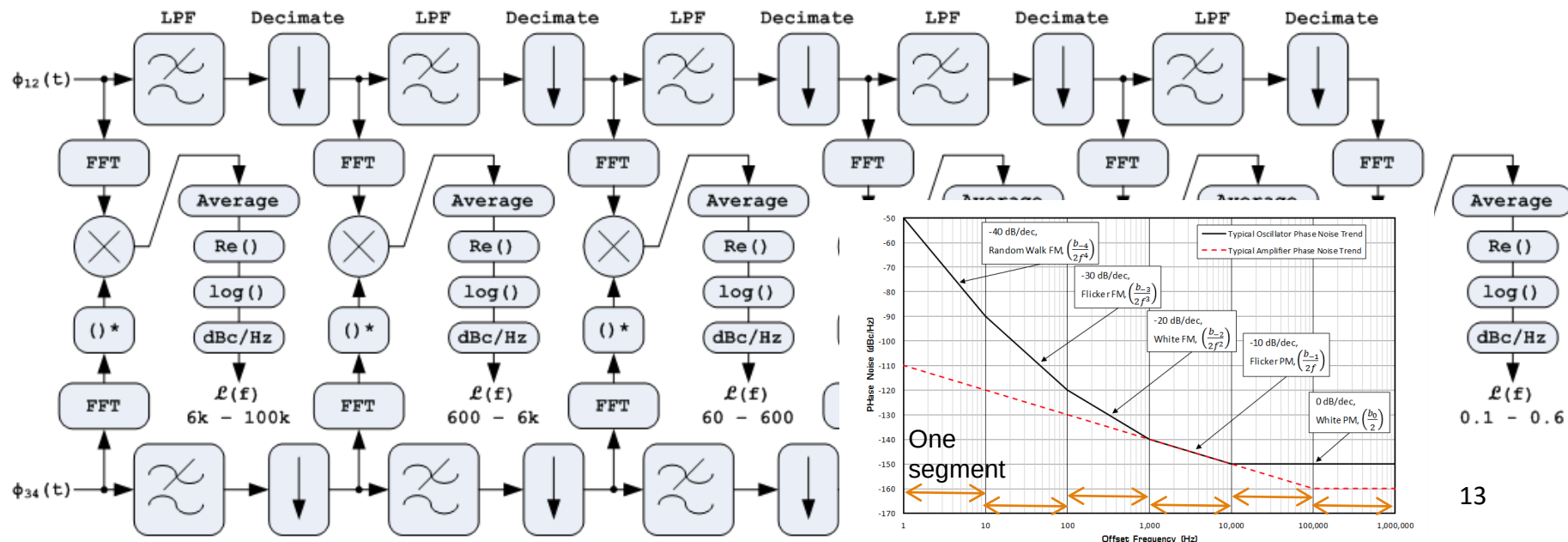
Cross correlation DUT-REF1 against DUT-REF2 to eliminate REF1 and REF2 by taking FFT, multiply and average

Problem: Due to the log frequency scale you need a multi decade FFT which becomes very slow



What is segmented FFT?

- First FFT covers a frequency decade
- Each LPF + Decimate creates input for next FFT
- Lower frequency FFT's run slower
- On log scale all decades have similar resolution
- Output of all FFT's joined to create a log scale X axis with constant resolution
- High frequency segments run often so good for reducing REF noise
- Feasible to cover 6 or more decades



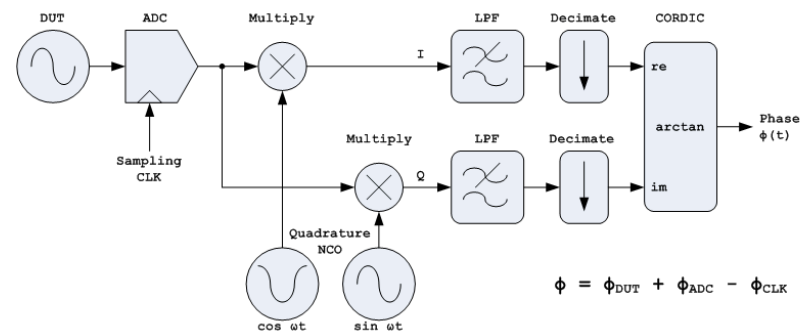
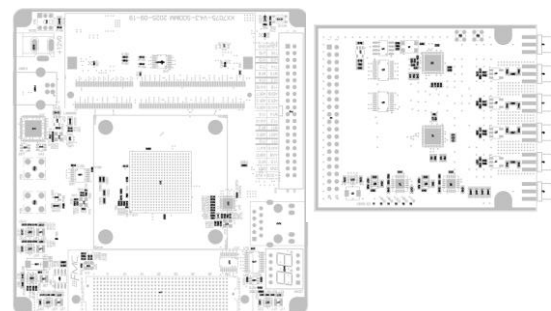
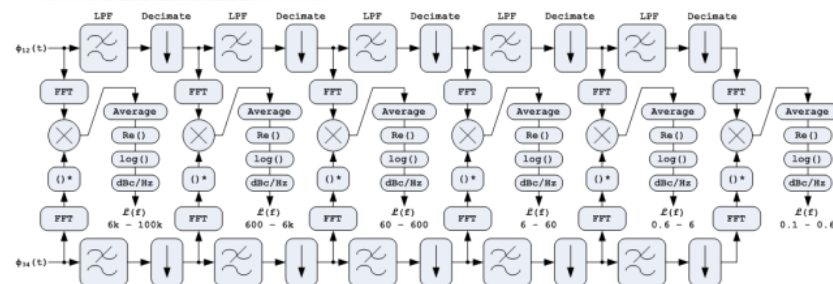
Translation to DSP (2)

- Why use ADC and I/Q mixing in digital domain?
 - All processing errors add up as additional phase noise.
 - A discrete I/Q mixer introduces I and Q phase errors.
 - PLL based LO adds phase noise even when locked to ADC clock
 - Discrete steep low pass filters have a lot of phase distortion
 - Impossible to have a high quality analog ATAN to calculate the phase
 - In the digital domain there are also some problems
 - Digital means there is quantization, both in time and resolution and this means possible loss of accuracy
 - Some operations are computational intensive.
 - A bit true simulation used to verify all calculations is very slow
 - But the required quality can be designed, proven and will never change

Mapping processing to hardware



- ADC (AD9623):
 - 4 inputs, 125 Msample/s, 14 bits
 - Use aliasing above Nyquist (above 62.5 MHz).
- Down conversion
 - Artix-7 FPGA with internal memory and lots of DSP cores
- Connection to PC
 - USB for control
 - UDP over Gbit Ethernet for data transport
- Segmented FFT and Cross correlation
 - All done in fast PC
 - Computation pipeline uses real-time tasks
 - Presentation as background process



HW limitations to be taken into account



- Non linearity of the ADC
 - This mixes the DUT with a noisy zero Hz and increases the phase noise. Measured phase becomes $DUT + (ADC \text{ non linearity})$
 - Solution: Measure DUT twice ($DUT + ADC1-REF1$ and $DUT + ADC2-REF2$)
 - Cross correlation will remove ADC and REF “noise” as these are uncorrelated
 - System now has 4 inputs ($DUT1, DUT2, REF1, REF2$)
- 14 bit resolution of the ADC
 - If ADC clock is very “close” to DUT/REF it causes quantization in the 10 kHz to 100 kHz FFT range
 - Solution: Shift ADC clock to non harmonic relation with DUT/REF
 - 125 MHz -> 137.778 MHz
- Impossible to have NCO perfectly tuned to DUT/REF
 - Causes linear trend of phase
 - Solution: De-trend phase data of each input
- Occasional loss of data due to UDP transport
 - Solution: Add Extensive loss detection and recovery
- System design done, from here it is just implementation.

Initial implementation approach

- All design/coding/testing to be done by an AI (Claude Fable/Opus) because of insufficient Verilog knowledge.
- Provide the AI with schematic of FPGA and ADC prototype HW.
- Provide tools: Vivado, GNU cross compiler. JTAG access to hardware
- Ask the AI: Design a dual subtractive phase noise analyzer using FPGA for phase measurement, PC for cross correlation and presentation, using Ethernet for data and USB for control.
- Wait till the AI tells it's ready.
- Test finished Phase Noise Analyzer Application

Getting frustrated

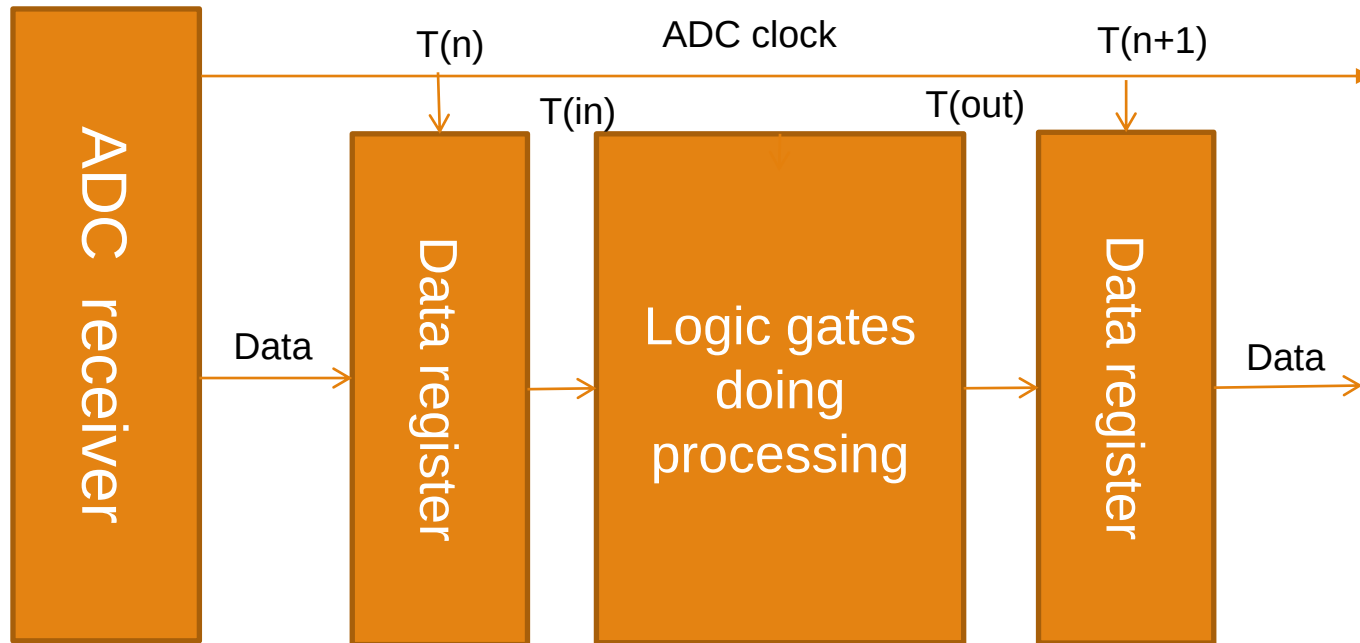
- Discover it does not work and never can work
- Start investigate architectural and design choices
- Tell the AI how to improve architecture and/or design.
- Wait till AI tells you everything is implemented
- Test and loop back

- Do this for some days till you have learned the limitations of AI

- Learning: The AI is not yet able to do serious Architecture work unless you give it the Architecture or you specify all boundary conditions you subconsciously take into account when designing a architecture

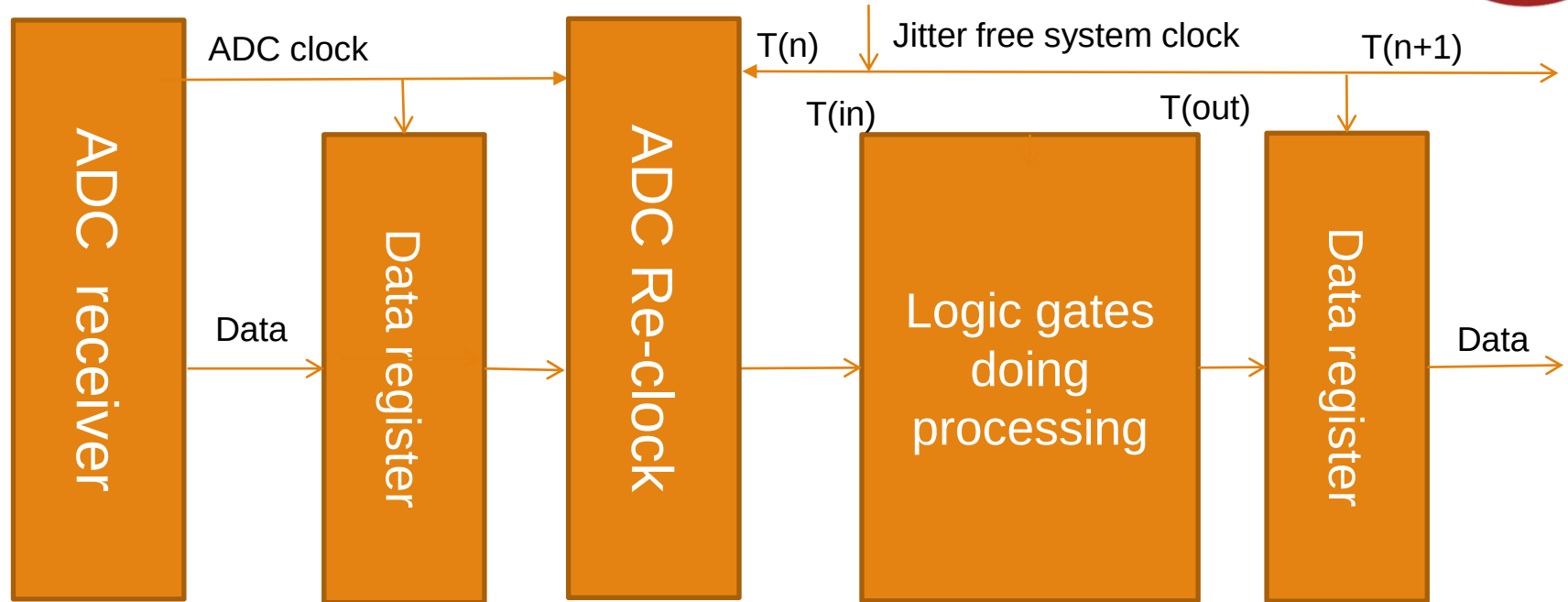
- Example:

Design created by AI



- ADC receiver recovers ADC clock and data and buffers using register
- Synthesis maps processing between registers such that $T(out) < T(n+1)$ assuming ADC clock has fixed period
- Actual recovered ADC clock has a lot of jitter (dT) causing occasional bit error because $T(out) + dT > T(n+1)$ because I did not tell the AI to create timing margin.

Suggest a more robust design



- Tell AI to Re-clock ADC data on jitter free system clock using small FIFO buffer
- Clock all processing on jitter free clock to remove timing uncertainty
- System clock faster than ADC clock to ensure timing margin.

Improved design approach

- Create a clean Architecture and Design document that the AI MUST adhere to
- Create strict instructions for the AI on how to review its own work against a large set of rules.
- Teach the AI how to create an incrementally testable SW system
- Teach the AI how to do units of work including planning, documenting, testing and committing
- Start from scratch while constantly checking if the AI does what it is supposed to do.
- Accept to be disappointed, do not get angry, improve rules and way of working, watch context size, switch between models to stay in budget and slowly get results
- This covers design, how about testing?

How to test the whole system?



- Mirror traditional analog system debugging:
 - Add taps along the whole DSP chain to observe each step (from ADC to input of segmented FFT)
 - Add synthetic oscillators in the FPGA that can simulate “any” ADC input
- Add scope and spectrum analyzer observers that can watch each tap and a signal generator to create test signals
- Instruct the AI to test each DSP block in an incremental approach (from start to finish for all chains) and avoid combinatorial explosion.
 - An AI will happily create thousands of test cases to test all possible combinations of the processing.
- Some examples of observing in between stages:

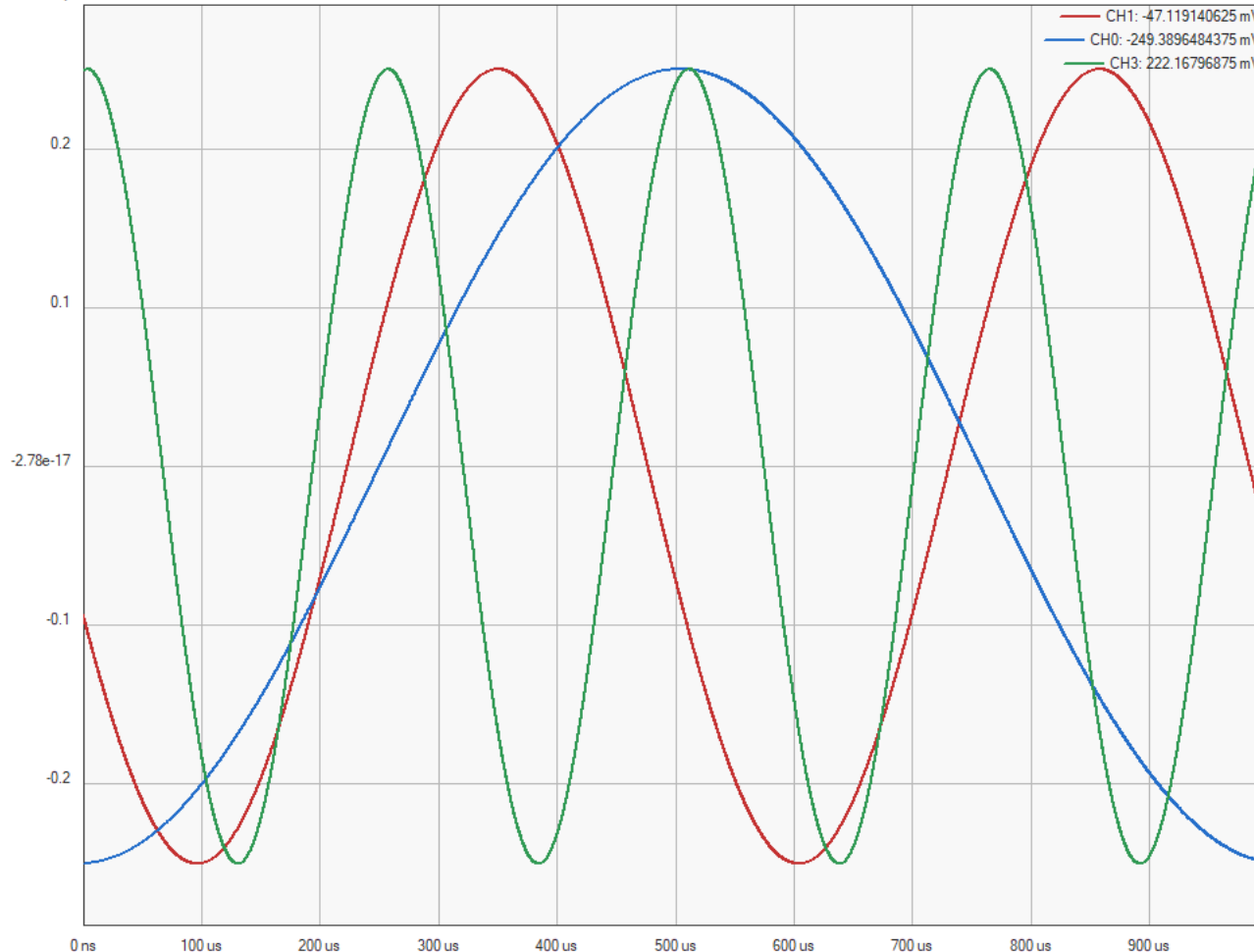
Scope observing signal generator output



4-Channel Oscilloscope

Timebase 1 ms

Y: Input Voltage (V) (linear)



SIGNAL PATH

Input Synthetic NCO

☐ Ext ref (SMA)

ADC clk (MHz) 137.7778

CORDIC rate maximum (4.306 MHz)

FPGA output ADC (raw)

VIEW

Mode Scope (time)

Timebase 1 ms

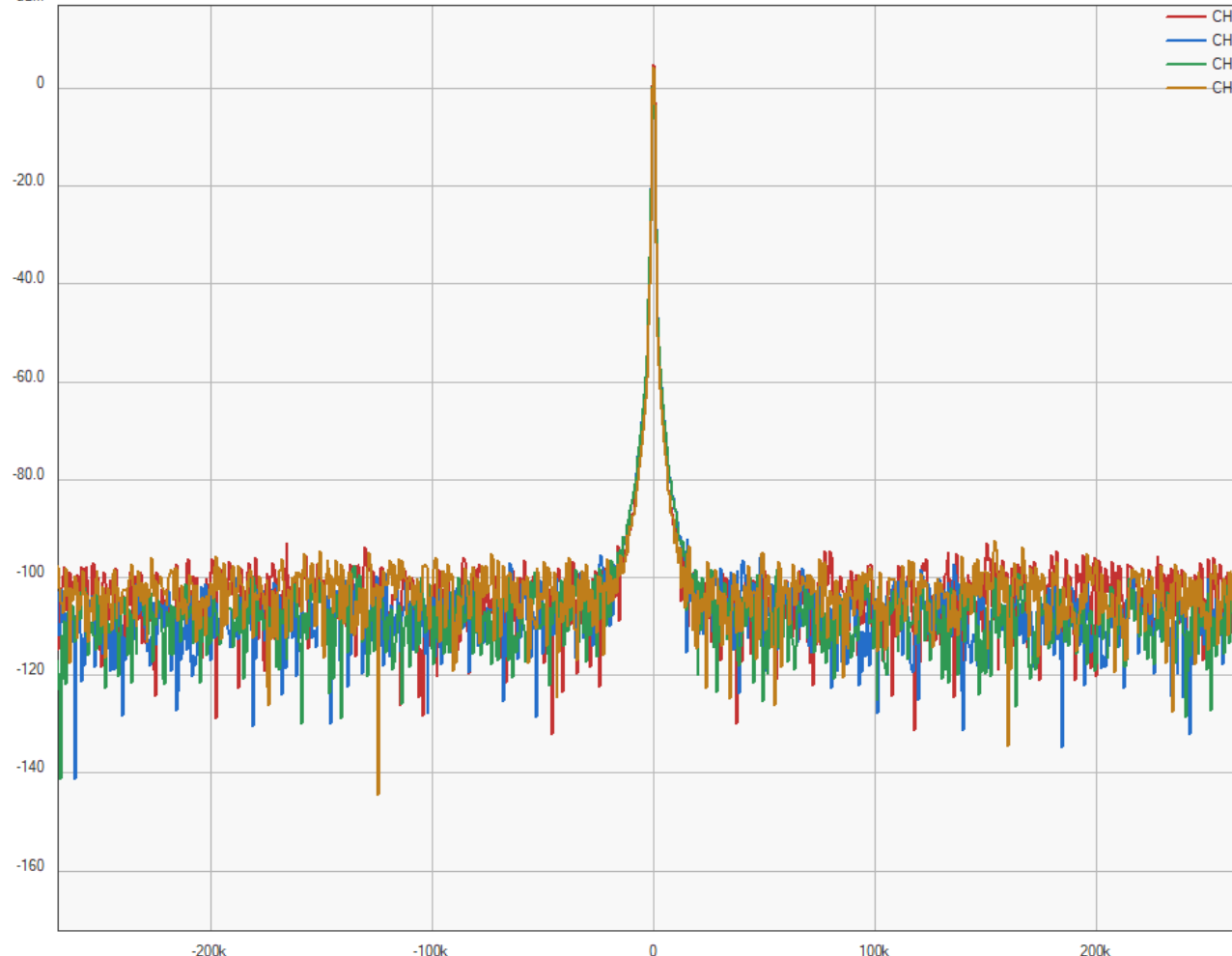
Scope display As-is

Spectrum analyzer observing down mixed signal

I/Q Wideband Spectrum

Input 1024 pt Hann 50% ovl avg off

dBm Y: Input Power (dBm) (linear)



SIGNAL PATH

Input Real ADC

☐ Ext ref (SMA)

ADC clk (MHz) 137.7778

CORDIC rate rate/8 (538.2 kHz)

FPGA output I/Q (pre-CORDIC)

VIEW

Mode FFT (frequency)

Reduce None (per-channel)

FFT length 1024

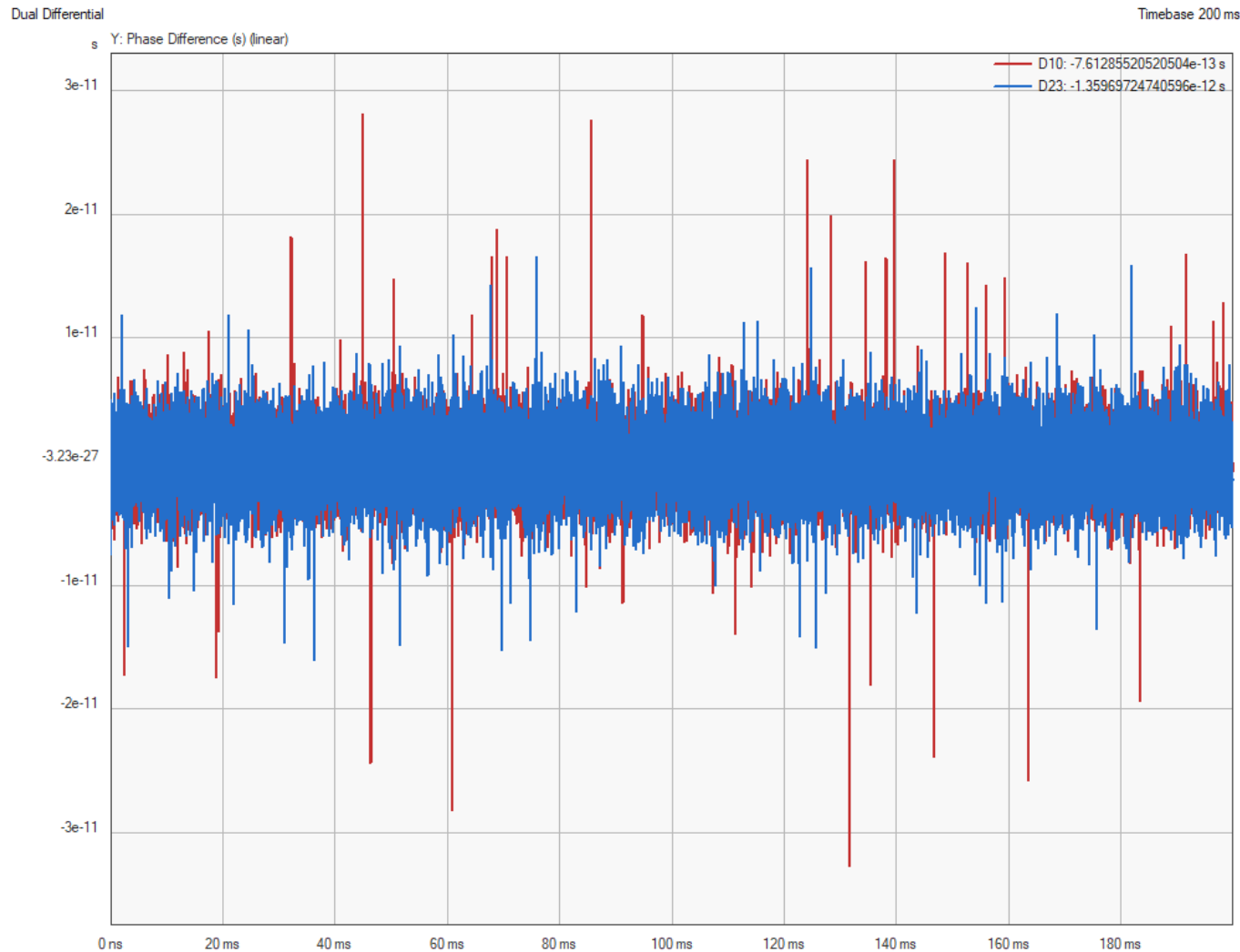
Fabric decades (S) 0 (host only)

☐ Segmented

☐ Average (overlapped)

Receive Off

Scope observing non-Gaussian phase errors



SIGNAL PATH

Input

☐ Ext ref (SMA)

ADC clk (MHz)

CORDIC rate

FPGA output

☐ Leakage corr.

Health slow rate (Hz)

☒ Detrend

Roles dA,dB,r1,r2 Target refs (t,r,r1,r2)

Samplerate

Max record: 5 s

Output

VIEW

Mode

Timebase

Scope display

**After a substantial
amount of hours:**

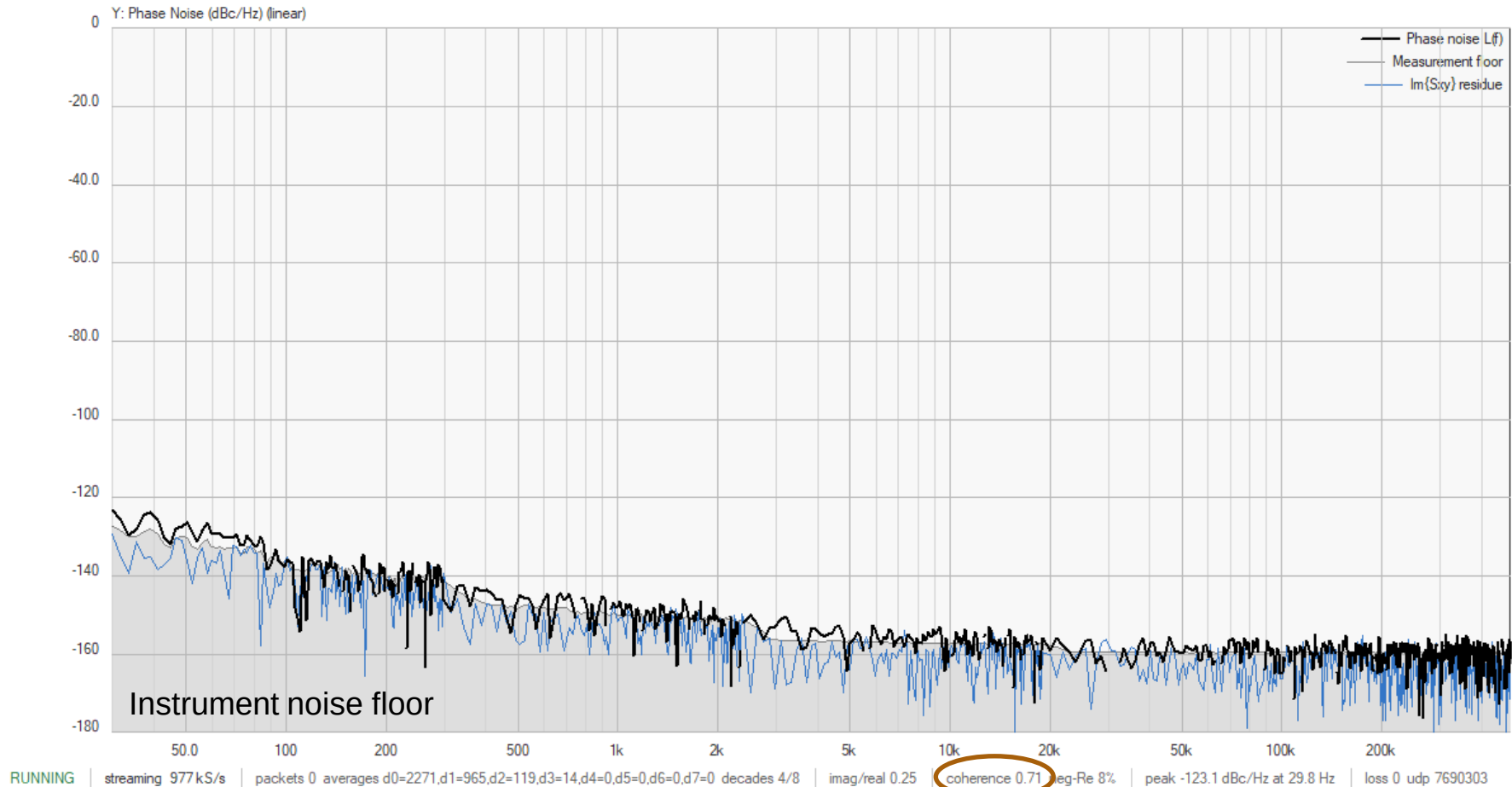
It's working! (4 decades)

Measuring synthetic source



Cross Correlated Phase Noise

Segmented 1024 pt Hann 50% ovl avg on



Improves over time



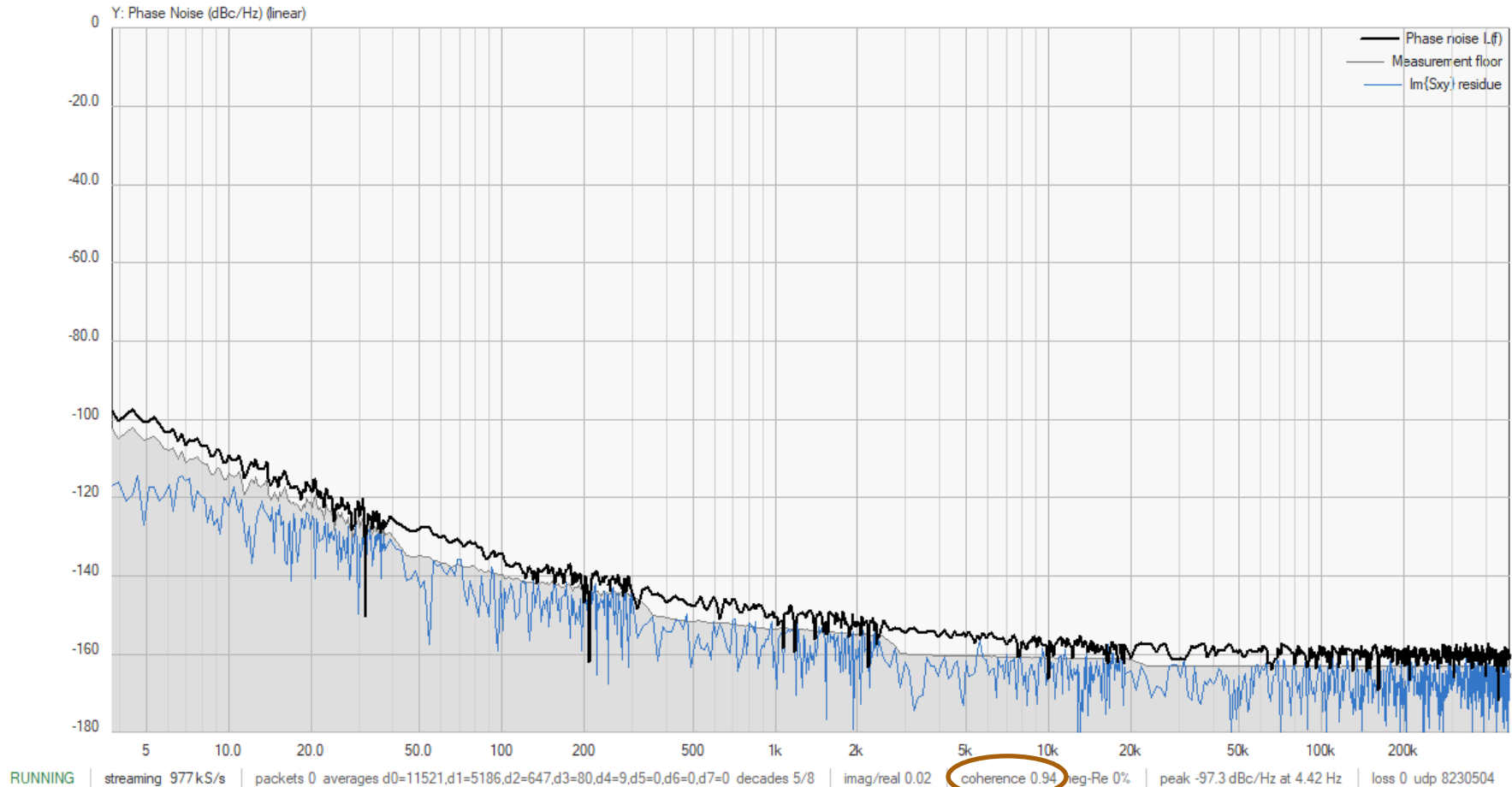
It's working! (5 decades)

Each additional decade takes 'decade ratio' more time



Cross Correlated Phase Noise

Segmented 1024 pt Hann 50% ovl avg on



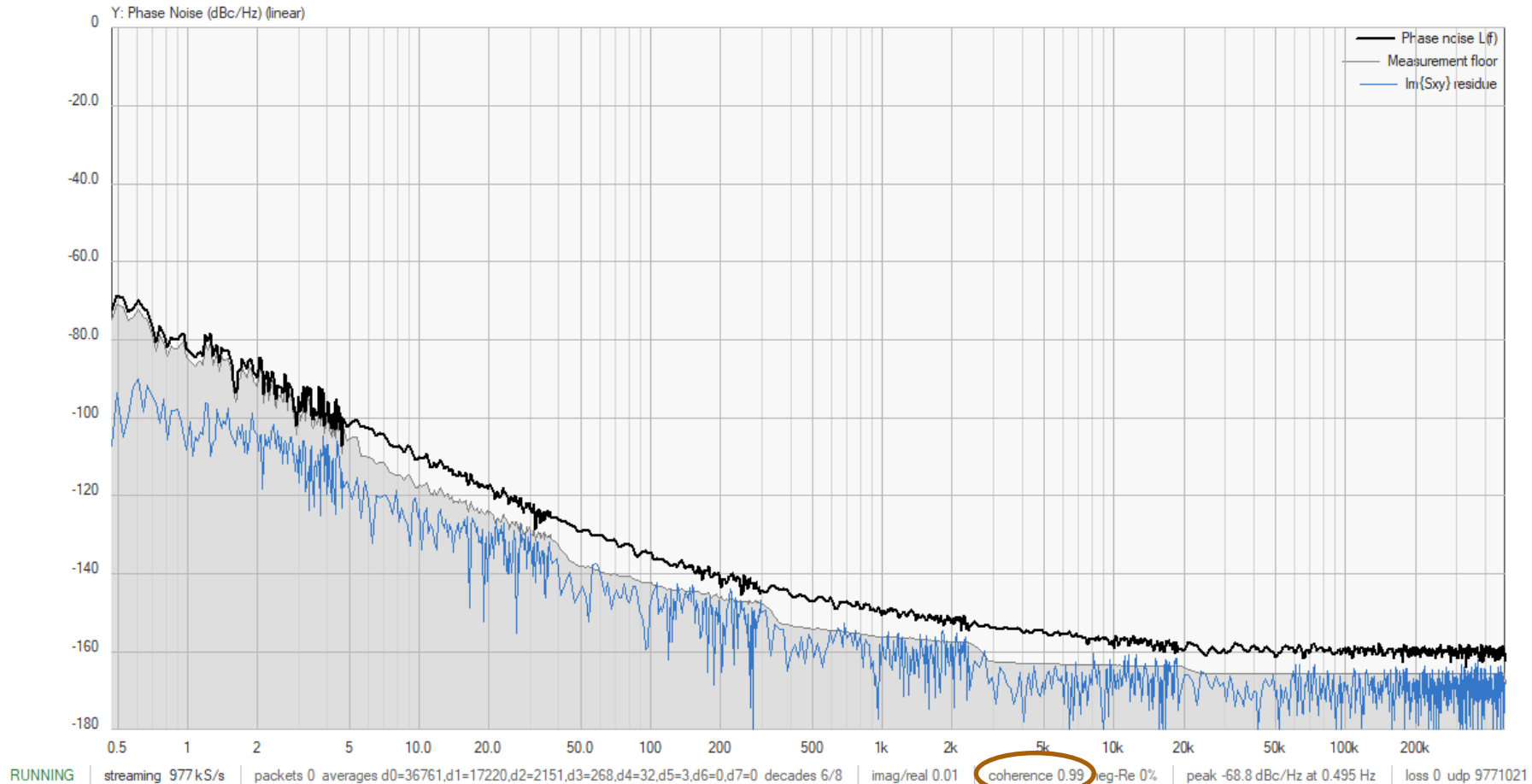
It's working! (6 decades)

Each additional decade takes 'decade ratio' more time



Cross Correlated Phase Noise

Segmented 1024 pt Hann 50% ovl avg on



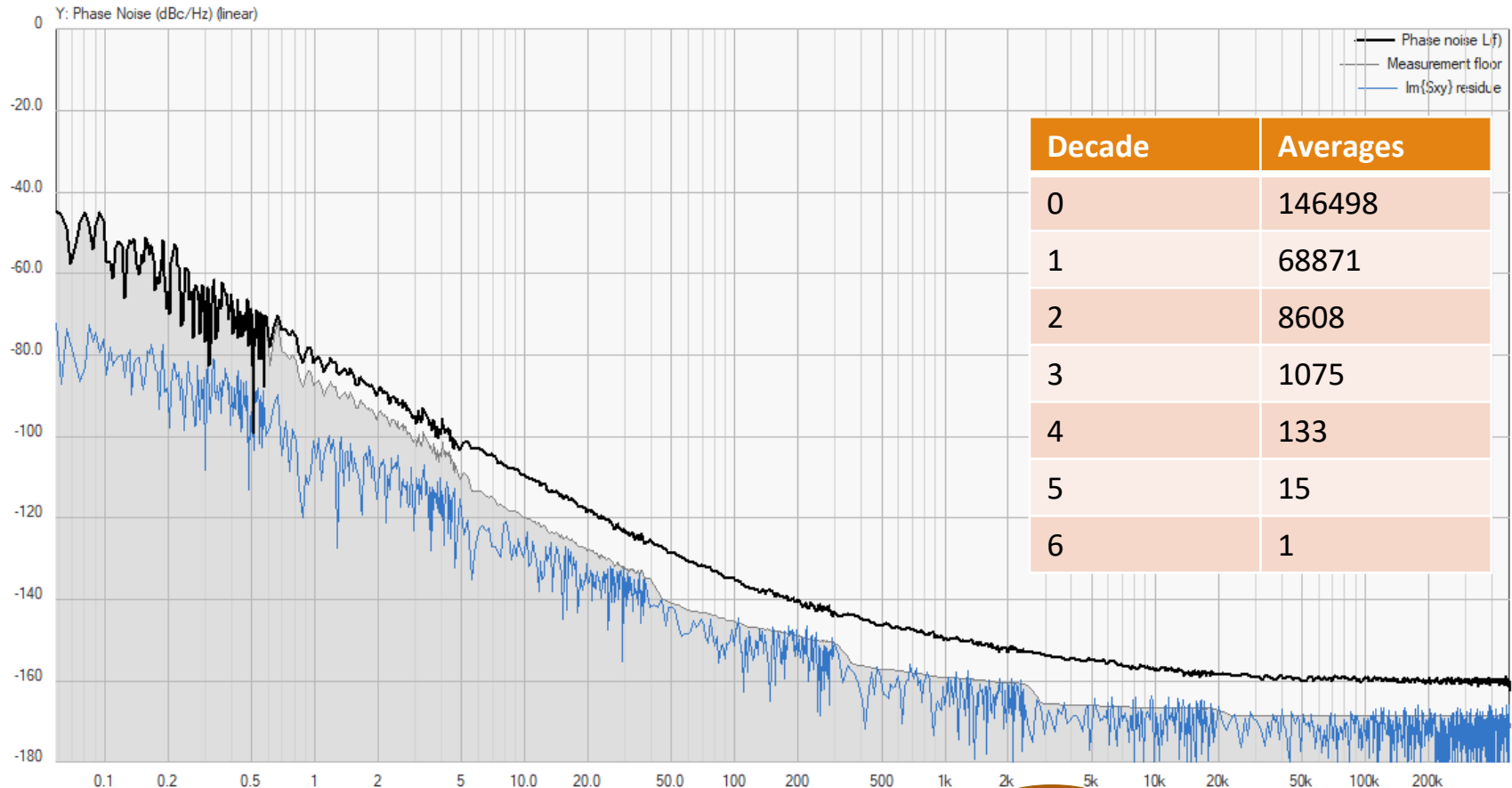
It's working! (7 decades)

Each additional decade takes 'decade ratio' more time



Cross Correlated Phase Noise

Segmented 1024 pt Hann 50% ovl avg on

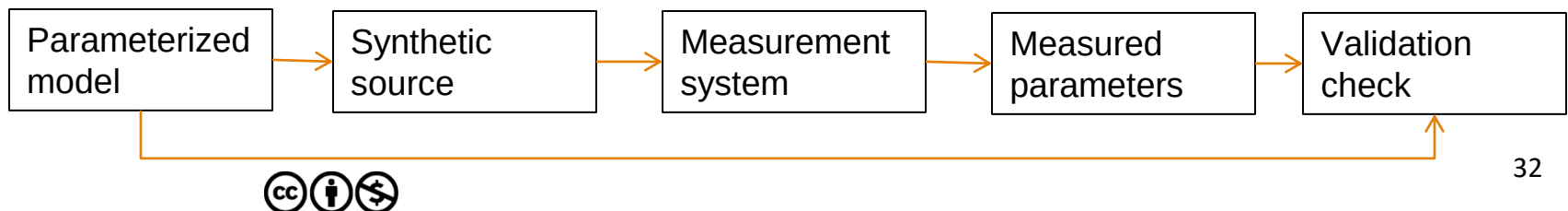


RUNNING | streaming 977 kS/s | packets 0 averages d0=146498,d1=68871,d2=8608,d3=1075,d4=133,d5=15,d6=1,d7=0 decades 7/8 | imag/real 0.00 | coherence 1.00 | reg-Re 0% | peak -44.6 dBc/Hz at 0.0582 Hz | ...

How to get confidence in the numbers?



- Both host-only and host+fpga can run in simulation mode.
- Simulation supports modeling of DUT+2xREF in two realistic usage scenarios:
 - Source phase noise model for $L(f)$ measurements
 - Source stability model for xDEV measurements
- host+fpga injects simulated sources right after ADC
- **Every measurement function has an associated test that can both prove and disprove the function using the simulated sources**



“Free” functionality

- Once you have all inputs in the digital domain you can use signal processing to add more functionality:
 - Scope: View phase versus time
 - Useful to check quality of inputs
 - Spectrum Analyzer: View power versus frequency
 - Useful to check quality of inputs (ground loops etc)
 - xDEV: View stability of Phase in many different ways
 - ADEV floor below $1\text{e-}13$ @ $\text{Tau} = 1\text{ s}$
- See next 2 slides for measurement examples
 - The legend shows which inputs were used

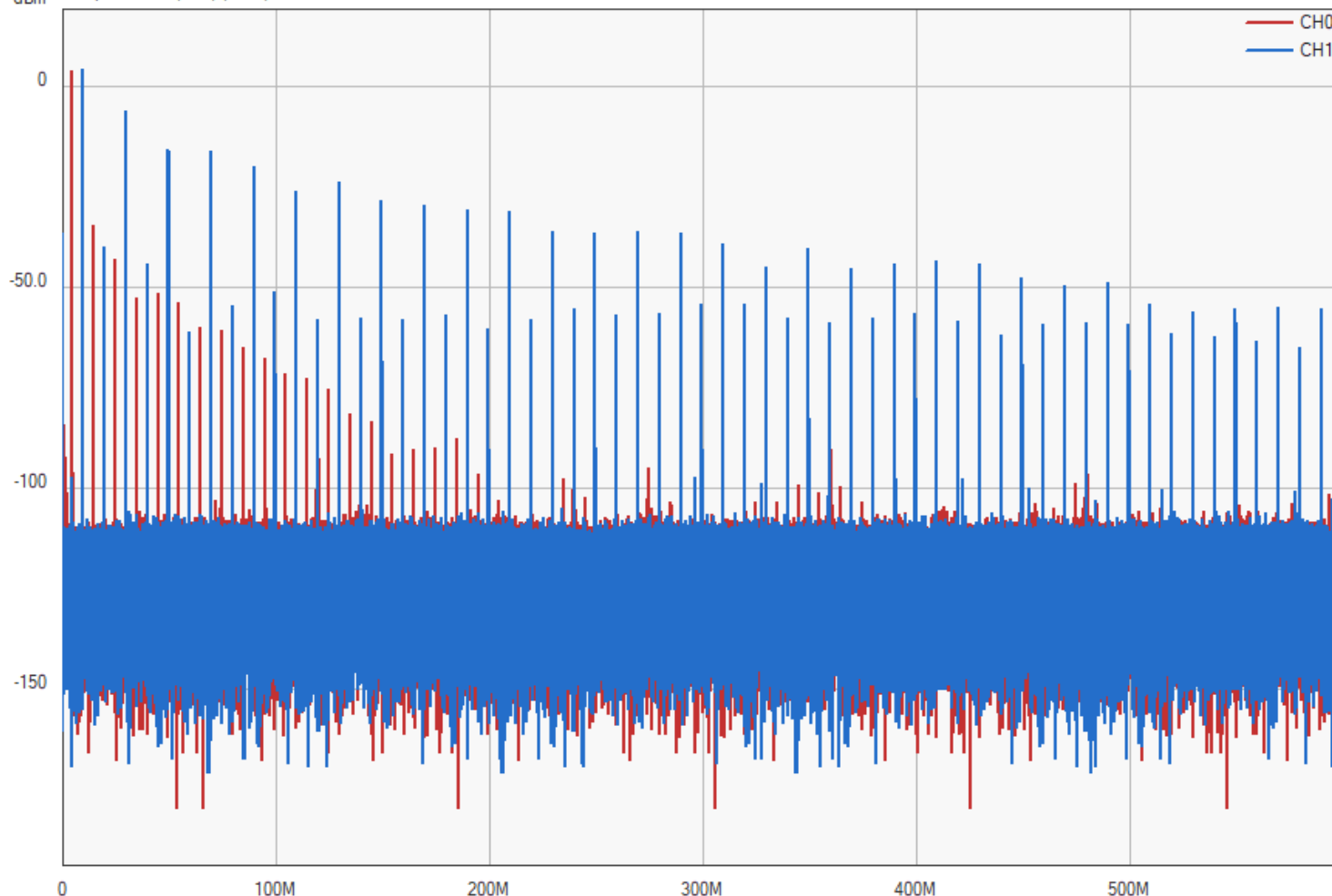
Wide band spectrum 0-600 MHz



Full Range FFT (aliased)

Segmented 1024 pt Hann 50% ovl avg on

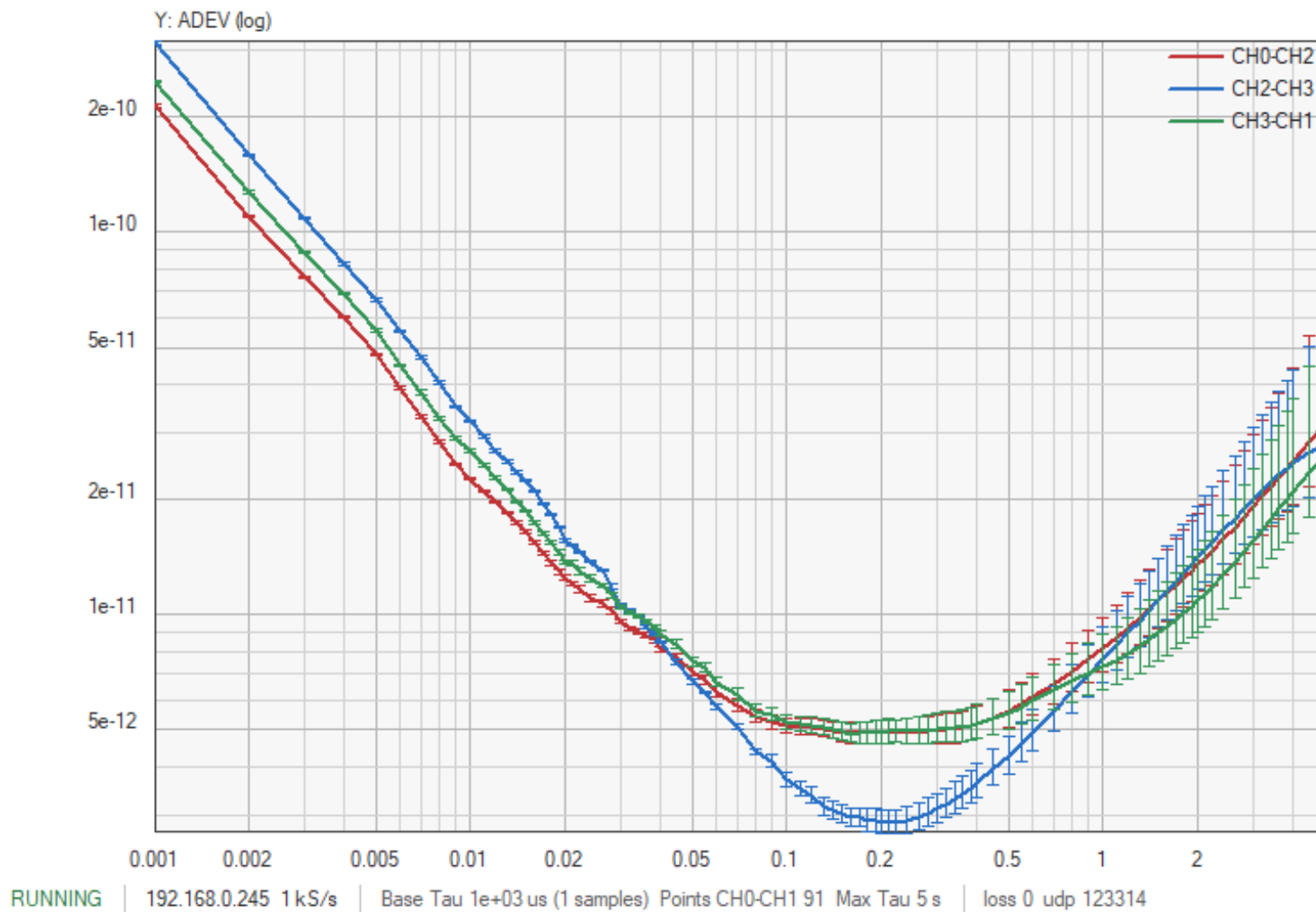
dBm Y: Input Power (dBm) (linear)



ADEV view

Tau ADEV

Timebase 1 s



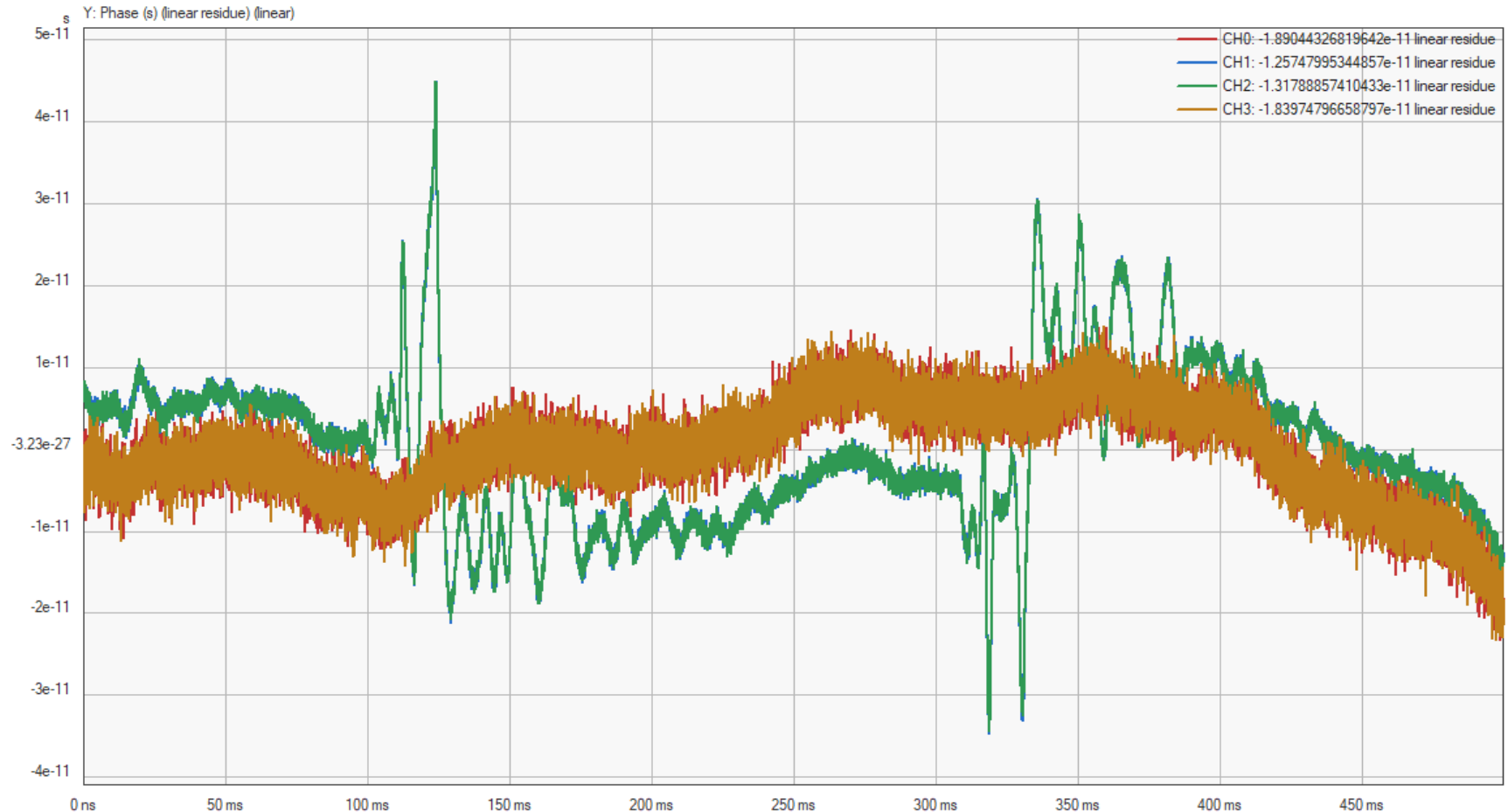
Diagnosing unstable Amplifier

Tiny phase disturbances clearly visible in phase view



4-Channel Oscilloscope

Timebase 500 ms linear residue



RUNNING | 192.168.0.245 1.08 MS/s | Rate 929 ns (1 samples) BW 538 kHz (point-sample, 1 of 1 averaged) Points CH0 16309 Span 500 ms | loss 0 udp 2957984

Ensuring correct measurements

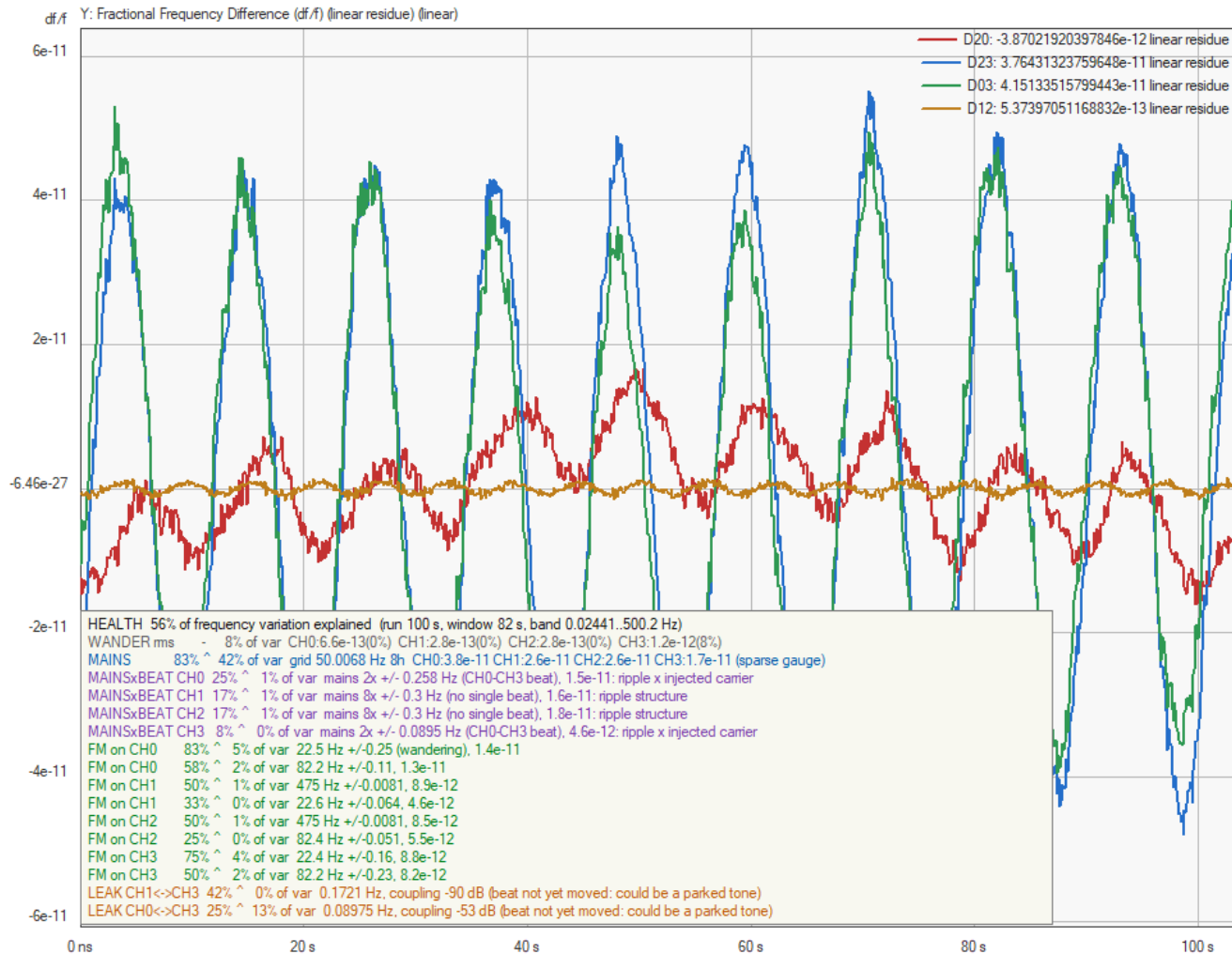


- Input signal problems:
 - Mains disturbance (50 Hz and harmonics)
 - Leakage between inputs (bad cables)
 - Drift
 - Exceeding ADC input range
- Internal limitations:
 - Leakage between inputs
- **How to detect these issues?**

Automatic Input Diagnostics: Detecting the source of disturbances

4-Channel Oscilloscope

Timebase 100 us linear residue



SIGNAL PATH

Input **Real ADC**

☐ Ext ref (SMA)

ADC clk (MHz) **137.7778**

CORDIC rate **rate/8 (538.2 kHz)**

FPGA output **Phase (CORDIC)**

☐ Leakage corr. **estart leakage learn**

Health slow rate (Hz) **25**

☒ Detrend

Roles dA,dB,r1,r2 **1,2,0,3** Pairs (d or a-b, $\hat{A}$ -4) **2-0,2-3,0-3,1-2**

Samplerate **10 Hz**

☐ Mains sync (<=100 S/s)

Max record: 23.1 d

Output **Frequency**

VIEW

Mode **Scope (time)**

Timebase **Forever (rolling)**

Scope display **Linear residue**

Some real Phase Noise measurements

- A fairly good 10 MHz OCXO
- Signal generator 10 MHz ref output
- Signal generator 10 MHz output
- Signal generator 100 MHz output
- tinySA 10 MHz output
- tinySA 600 MHz output

A fairly good 10 MHz OCXO

1 Hz : -100 dBc/Hz, 10 kHz : -160 dBc/Hz

RF
Seminar

Cross Correlated Phase Noise

Segmented 1024 pt Hann 50% ovl avg off

dBc/Hz Y: Phase Noise (dBc/Hz) (linear)



RUNNING | streaming 1.08 MS/s | packets 0 averages d0=1294219,d1=180943,d2=22396,d3=2701,d4=294,d5=18,d6=0,d7=0 decades 6/8 | imag/real 0.08 | coherence 0.95 neg-Re 1% | ADC lines n*16.087 kHz: 8 masked | ...

Same OCXO through not so good distribution amplifier



Cross Correlated Phase Noise

Segmented 1024 pt Hann 50% ovl avg on

dBc/Hz Y: Phase Noise (dBc/Hz) (linear)



Signal generator 10 MHz ref out

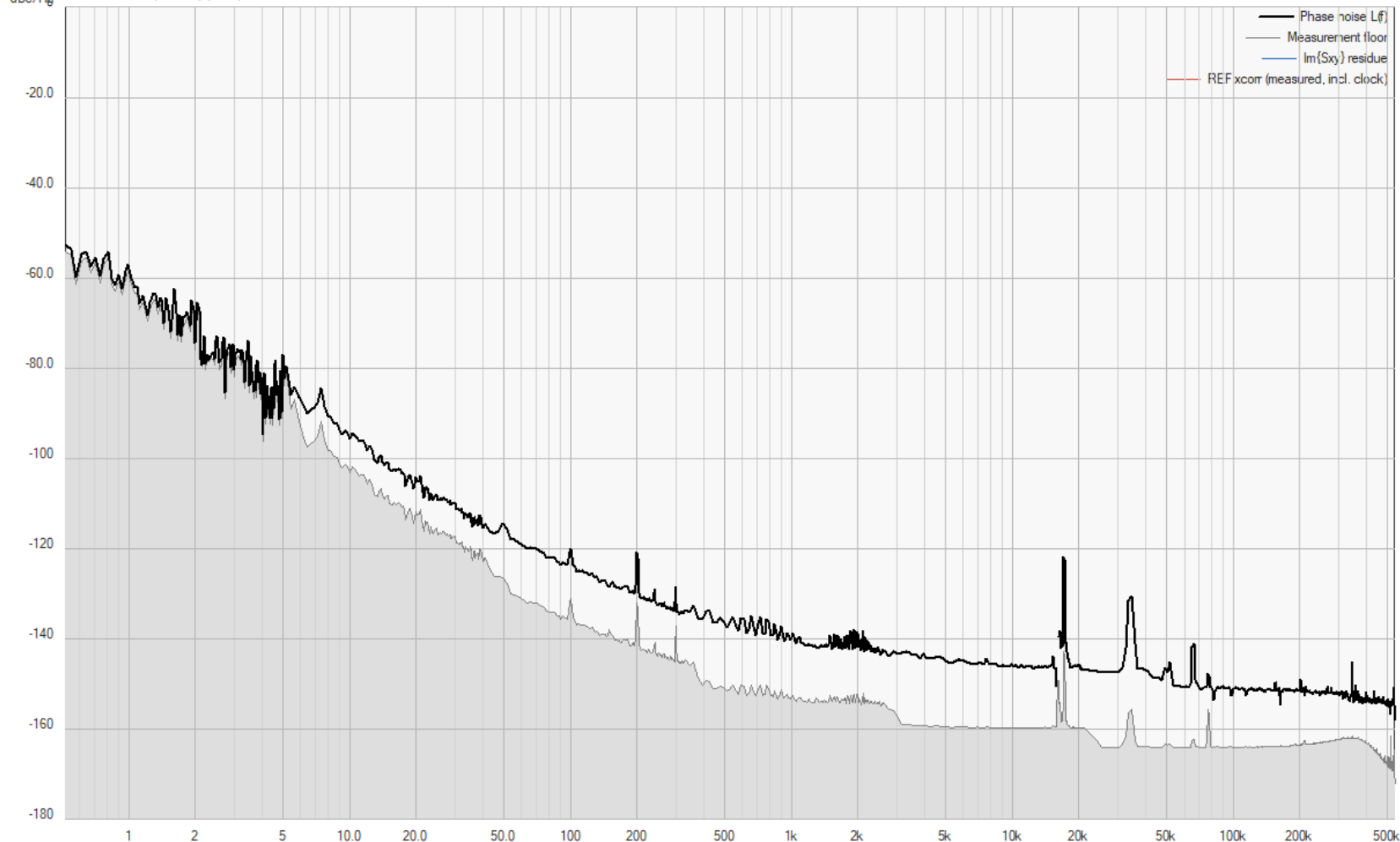
TCXO : 1 Hz : -60 dBc/Hz, 10 kHz : -145 dBc/Hz

RF
Seminar

Cross Correlated Phase Noise

Segmented 1024 pt Hann 50% ovl avg off

dBc/Hz Y: Phase Noise (dBc/Hz) (linear)



RUNNING | streaming 1.08 MS/s | packets 0 averages d0=130799,d1=18003,d2=2248,d3=279,d4=33,d5=2,d6=0,d7=0 decades 6/8 | imag/real 0.00 | coherence 0.99 neg-Re 0% | peak -52.7 dBc/Hz at 0.513 Hz | ...

Signal generator 10 MHz output

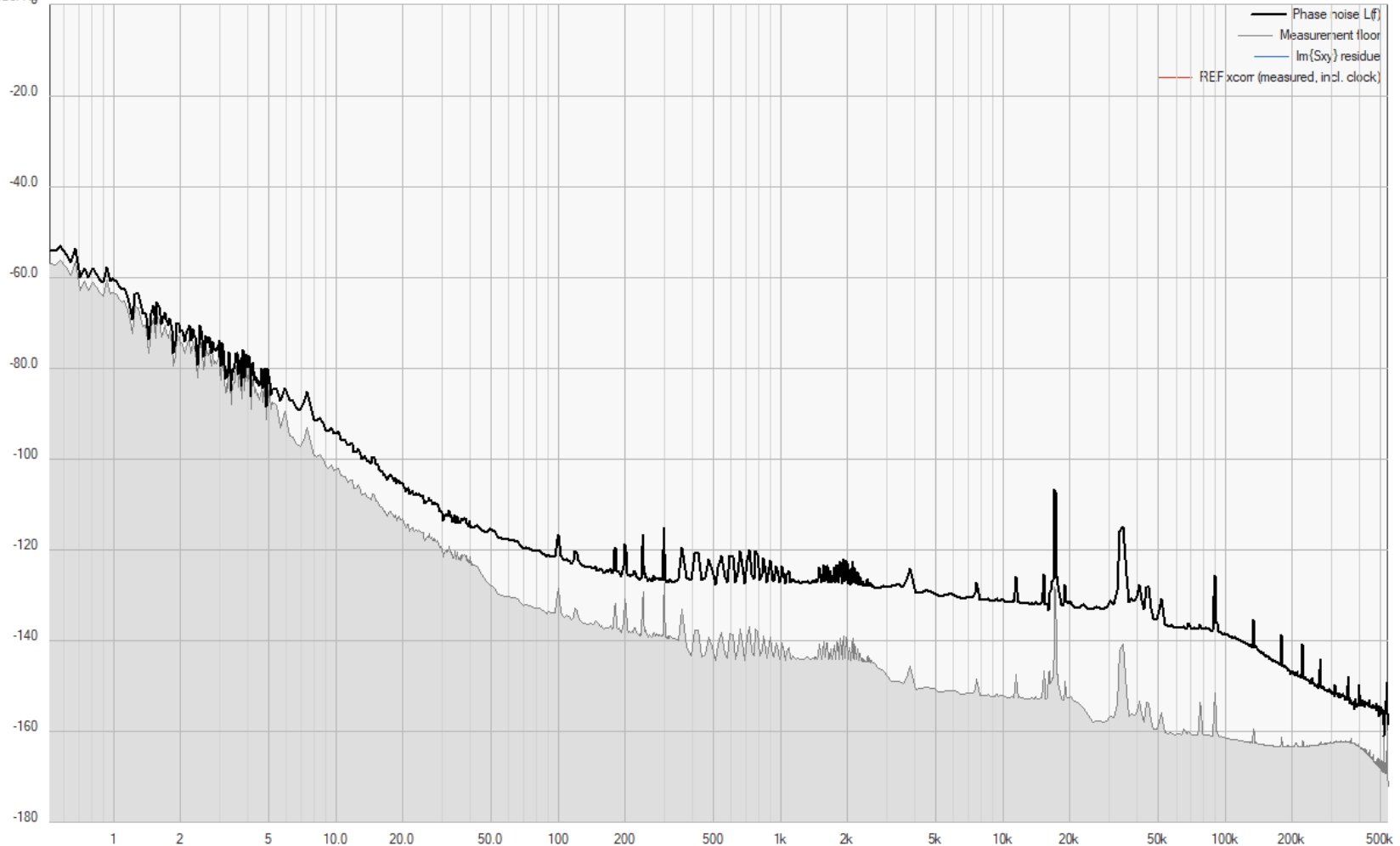
PLL : 1 Hz : -60 dBc/Hz, 10 kHz : -130 dBc/Hz



Cross Correlated Phase Noise

Segmented 1024 pt Hann 50% ovl avg off

dBc/Hz Y: Phase Noise (dBc/Hz) (linear)



RUNNING | streaming 1.08 MS/s | packets 0 averages d0=160504,d1=22179,d2=2771,d3=345,d4=42,d5=4,d6=0,d7=0 decades 6/8 | imag/real 0.00 | coherence 1.00 neg-Re 0% | peak -53.2 dBc/Hz at 0.577 Hz | ...

Signal generator 100 MHz output

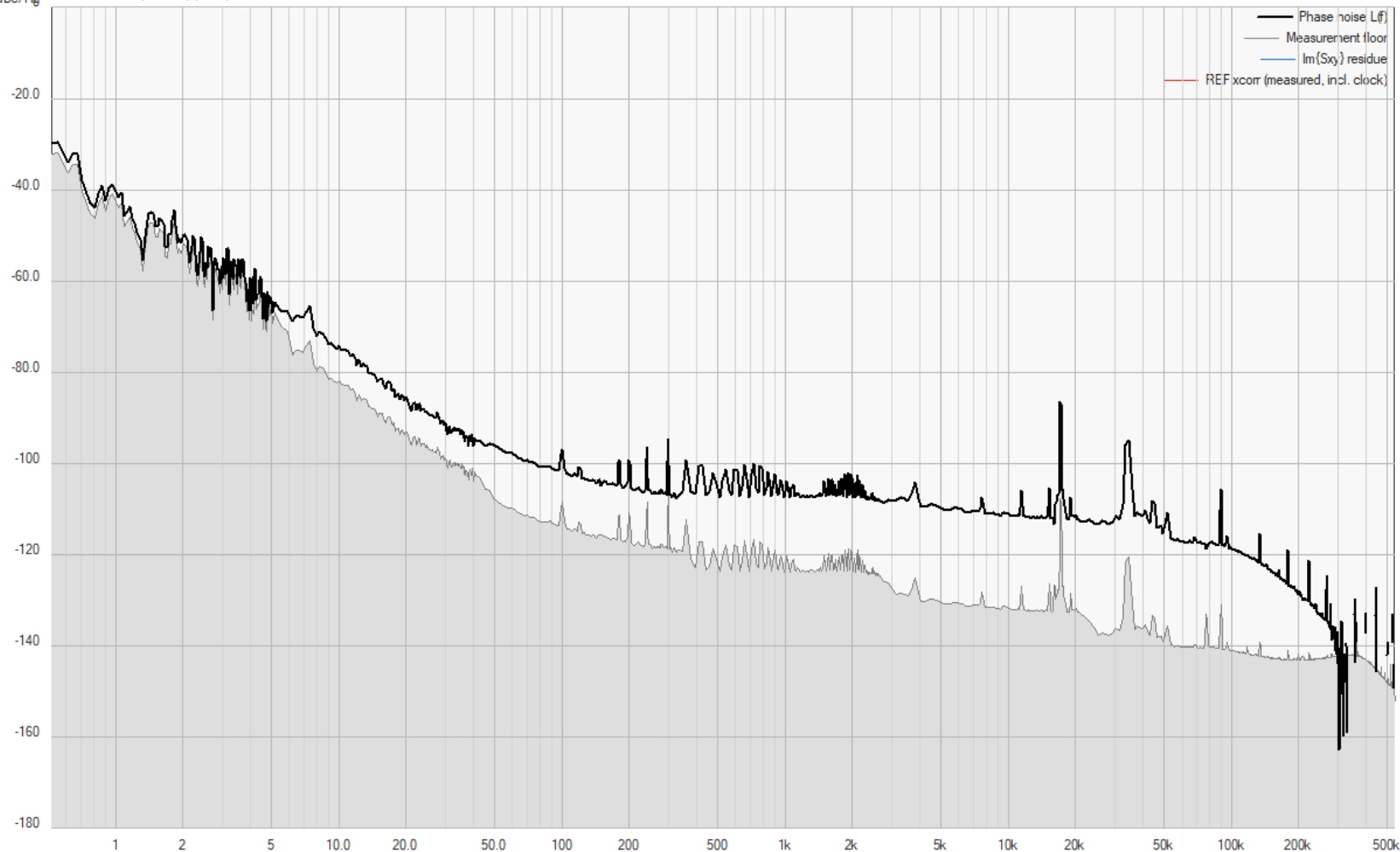
PLL : 1 Hz : -40 dBc/Hz, 10 kHz : -110 dBc/Hz (As predicted 20 dB worse)

RF
Seminar

Cross Correlated Phase Noise

Segmented 1024 pt Hann 50% ovl avg off

dBc/Hz Y: Phase Noise (dBc/Hz) (linear)



RUNNING | streaming 1.08 MS/s | packets 0 averages d0=132595,d1=18286,d2=2284,d3=284,d4=34,d5=3,d6=0,d7=0 decades 6/8 | imag/real 0.00 | coherence 0.70 neg-Re 13% | peak -29.6 dBc/Hz at 0.545 Hz | ...

tinySA Output at 10 MHz PLL :

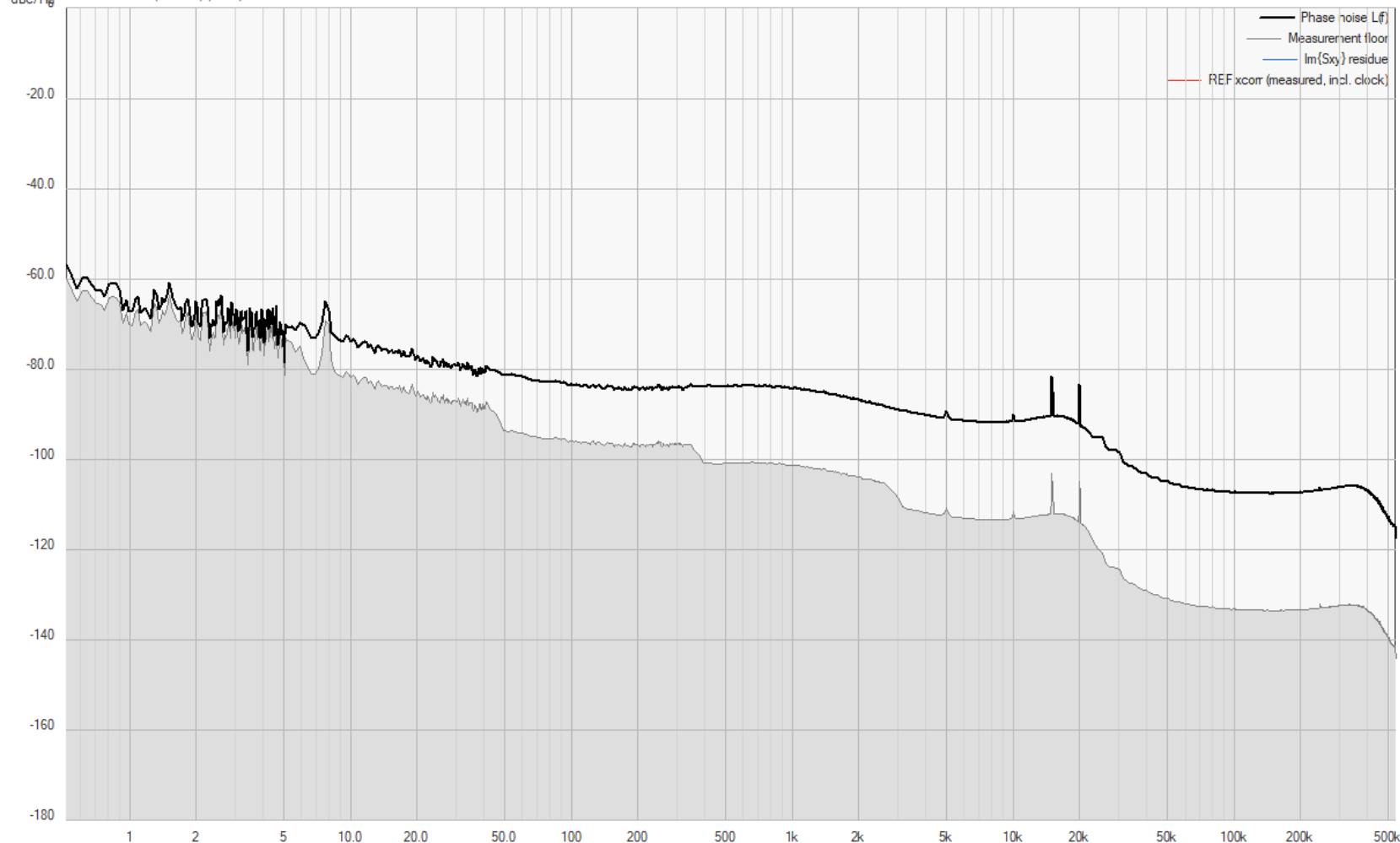
1 Hz : -65 dBc/Hz, 10 kHz : -90 dBc/Hz

RF
Seminar

Cross Correlated Phase Noise

Segmented 1024 pt Hann 50% ovl avg off

dBc/Hz Y: Phase Noise (dBc/Hz) (linear)



RUNNING | streaming 1.08 MS/s | packets 0 averages d0=158568,d1=21791,d2=2723,d3=339,d4=41,d5=4,d6=0,d7=0 decades 6/8 | imag/real 0.04 | coherence 1.00 neg-Re 0% | peak -56.8 dBc/Hz at 0.513 Hz | ...

tinySA at 600 MHz

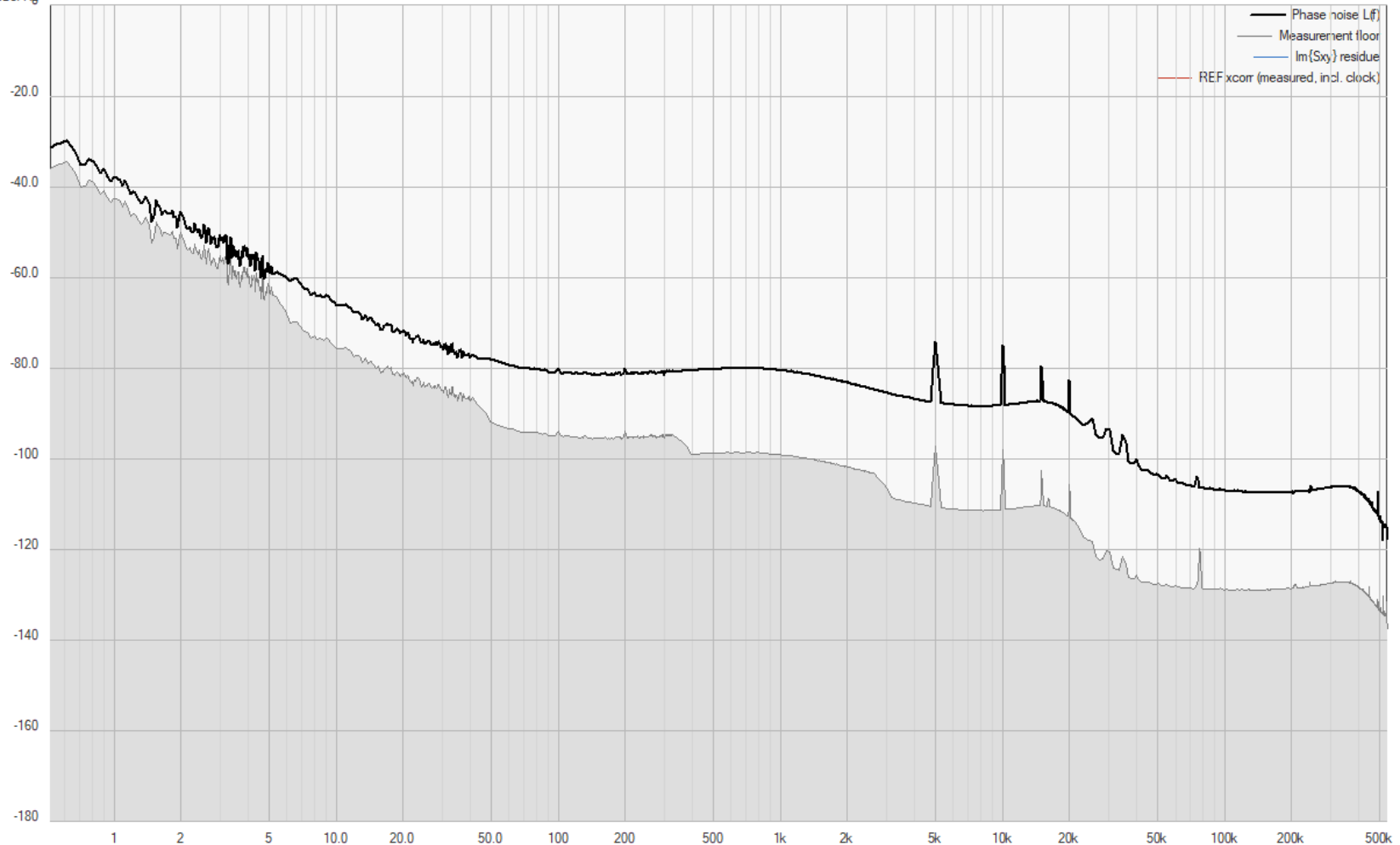
1 Hz : -40 dBc/Hz, 10 kHz : -85 dBc/Hz



Cross Correlated Phase Noise

Segmented 1024 pt Hann 50% ovl avg off

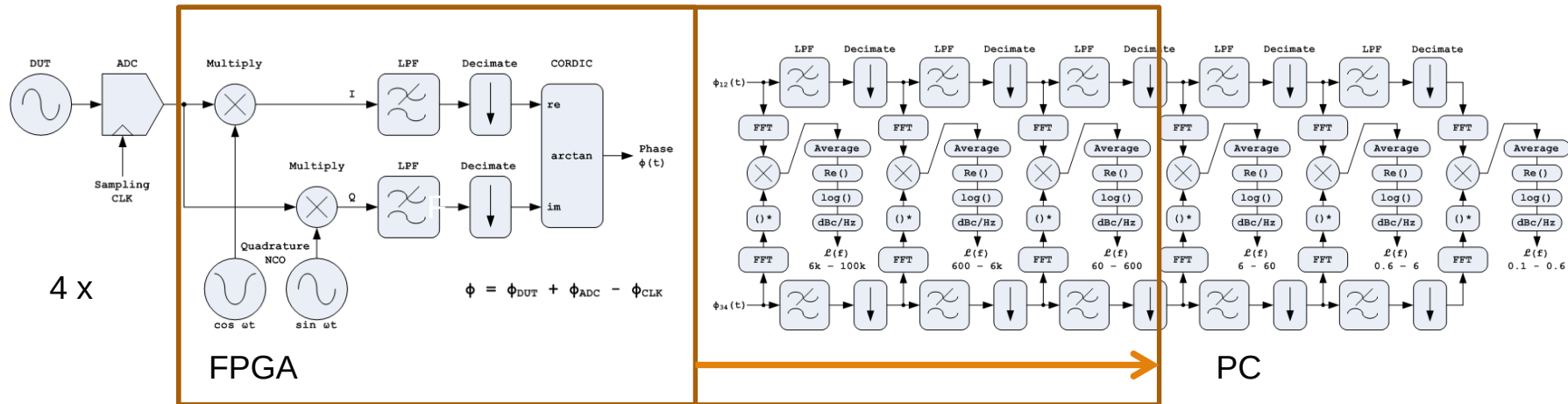
dBc/Hz Y: Phase Noise (dBc/Hz) (linear)



Further improvements:

- Smaller FPGA
 - Remove duplication
 - Downstream FFT segments share the same HW
 - Smarter DSP
 - Selected decimation strongly impacts complexity
 - Reduce bit width where possible
 - Bit width reduction can increase noise
 - Every reduction must be validated using synthetic (e.g. perfect) inputs against the -180 dBc/Hz target
- No Ethernet link
 - Reduce data link speed requirements

Reducing data link speed



- Dedicated 1 Gbit Ethernet link not always available
- Reduce data link rate by shifting processing to FPGA
- At most 3 FFT segments inside FPGA at cost of slower screen updates (0.3 Hz)
- Transport via USB 2.0 only
- Fully functional for most important measurement functions.

Reducing FPGA size

- Unfortunately, reducing data link speed increases the FPGA size
- The AI has some extra capability limitations:
 - By default it uses the textbook solution, even if computationally inefficient.
 - The AI is never lazy so it will easily generate a lot of functional duplication when doing incremental design
- Learning:
 - Tell the AI the long term goal is to use smaller FPGA
 - Provide the AI with direct instructions on how to setup the DSP chain to enable smaller FPGA size.
 - Check everything the AI does and question any design decision
- Intermediate result: Factor 2 reduction which compensates for the shift of processing into the FPGA

Future option: Extending to High Frequency DUTs



- Currently direct input up to 600 MHz
- Measuring higher input frequencies:
 - Down convert twice, each with own independent LO
 - Feed down convert inputs to the 2 DUT inputs.
 - Cross correlation will remove phase noise from LO PLLs in the same way as it removes the ADC linearity noise
- This is mostly a HW extension with limited SW impact

Conclusion

- The Phase Noise Analyzer has become a one stop tool to analyze oscillators!
- Using AI to implement systems of this complexity is still not straightforward
- I learned a lot about DSP and FPGA
- Currently no plans to make this into a product
 - The fun part is done, the remaining is hard work.
 - The required HW design (6 layer PCB with BGA) is currently beyond my capabilities.
 - Some remaining complex design challenges:
 - Reduction of leakage between the 4 inputs
 - Ability to measure below -175 dBc/Hz not yet proven.