Using GNU Radio to Explore the Consequences, Limits, and Behavior of DSA Systems

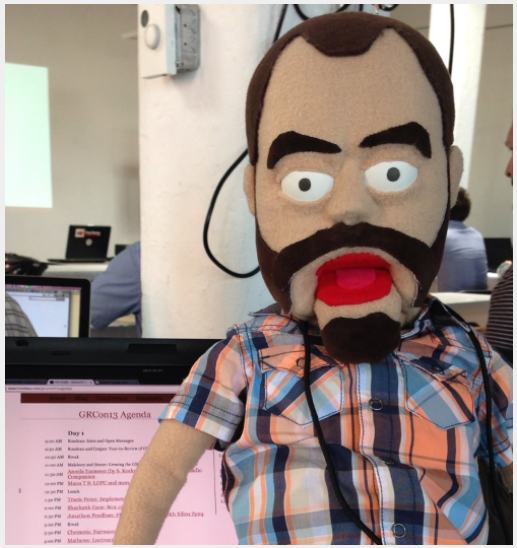
Tom Rondeau & Matt Ettus

(tom@trondeau.com, matt@ettus.com)

2014-04-01

Tom Rondeau

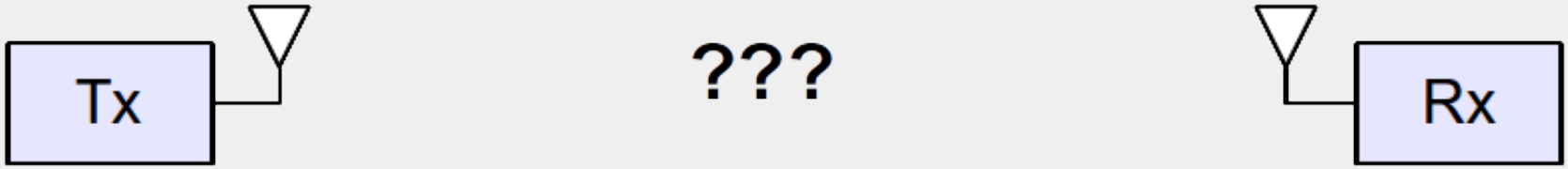
- Maintainer and lead developer for GNU Radio
- Consults through Rondeau Research
- Visiting Researcher at UPenn
  - (working with Jonathan Smith)
- www.trondeau.com
- gnuradio.org



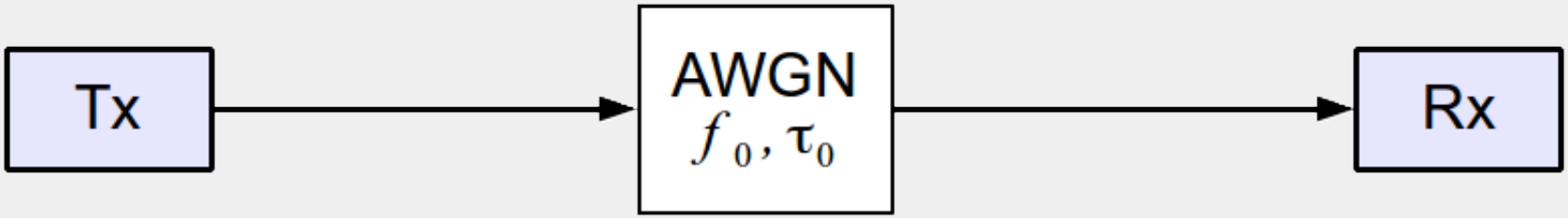
Matt Ettus

- Founder and President of Ettus Research
- Long time contributor to GNU Radio
- A National Instruments Distinguished Engineer
- www.ettus.com

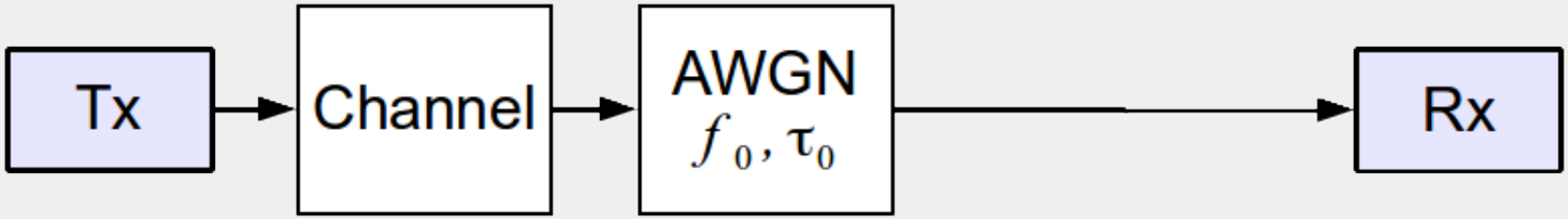
Outline



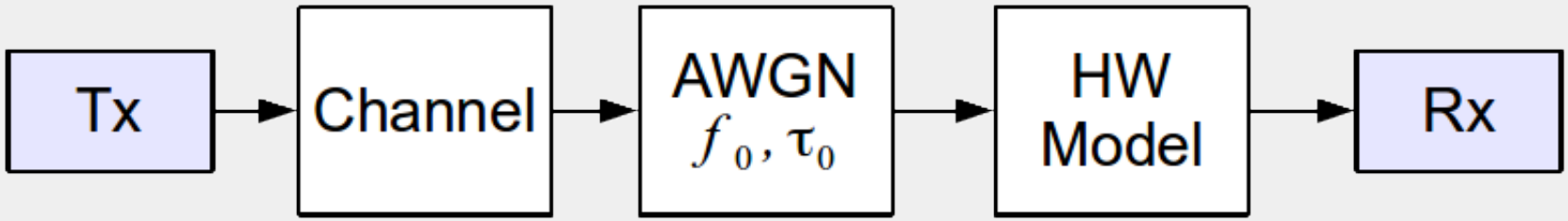
Outline



Outline

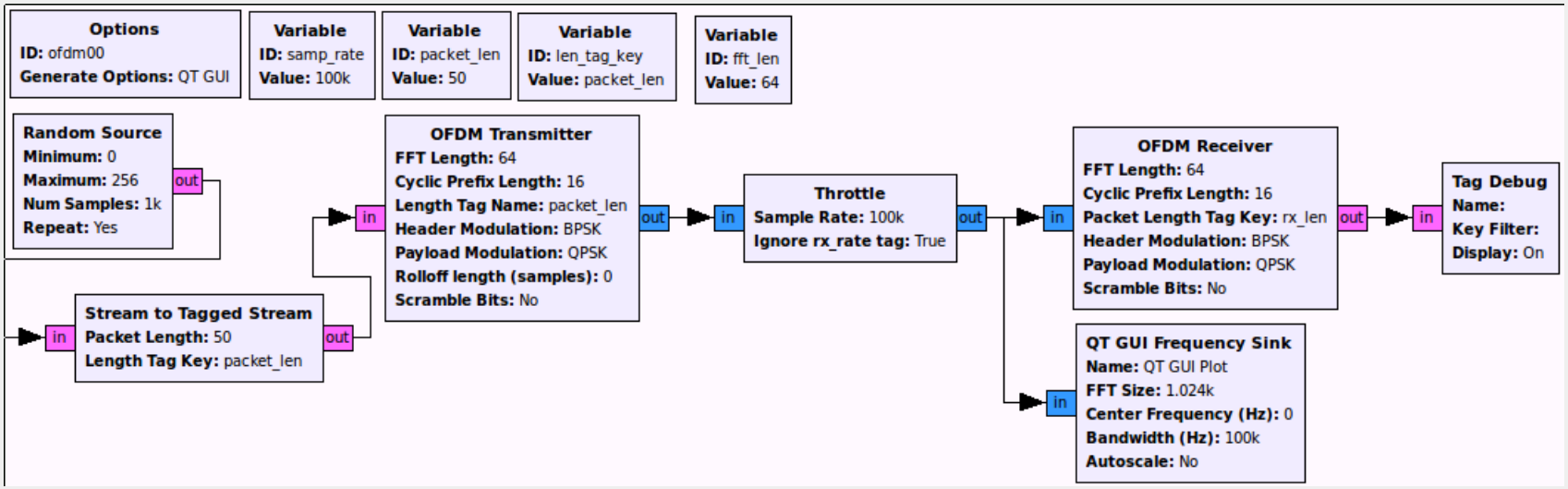


Outline

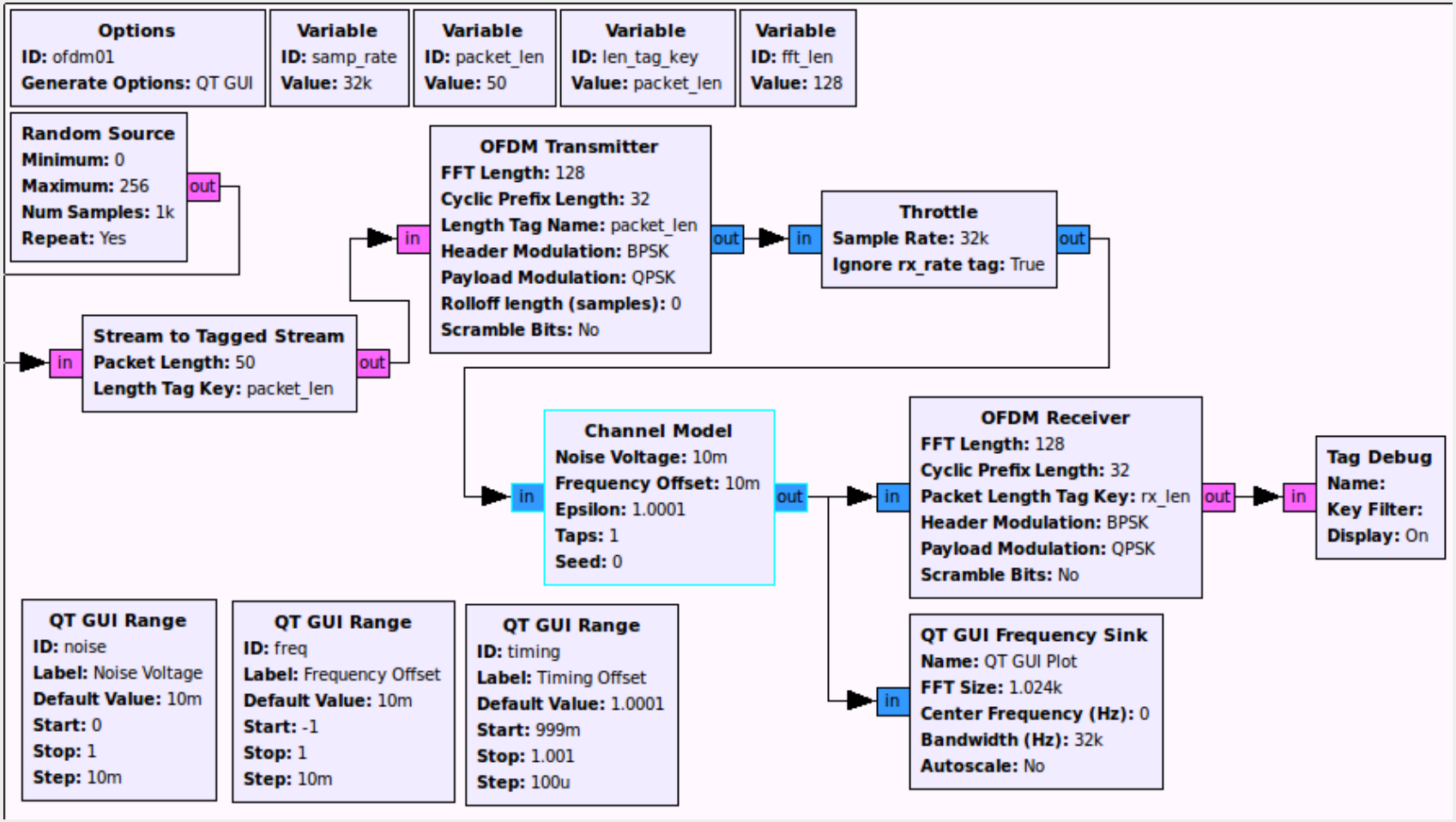


Outline

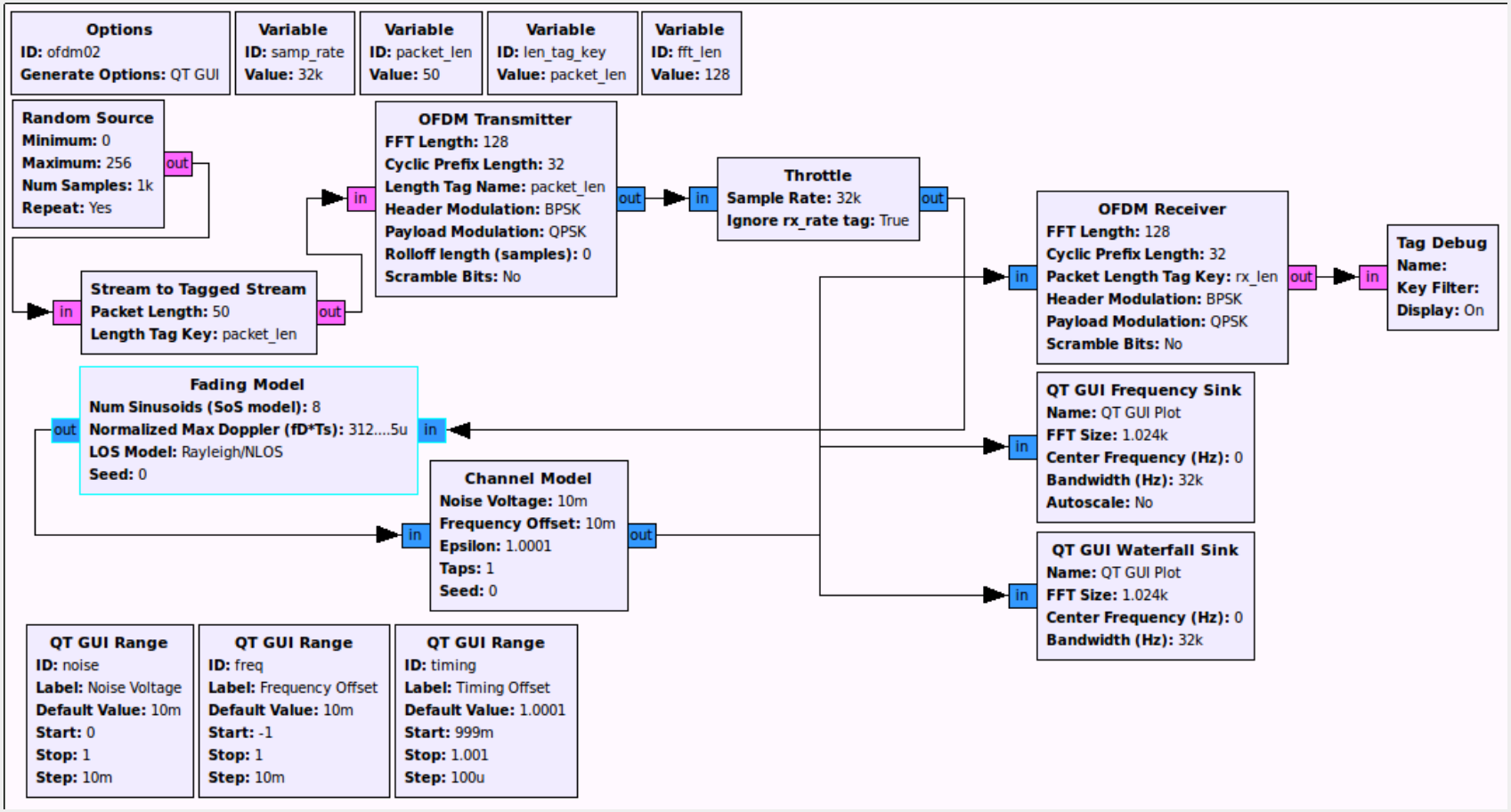
Verify it works without a channel



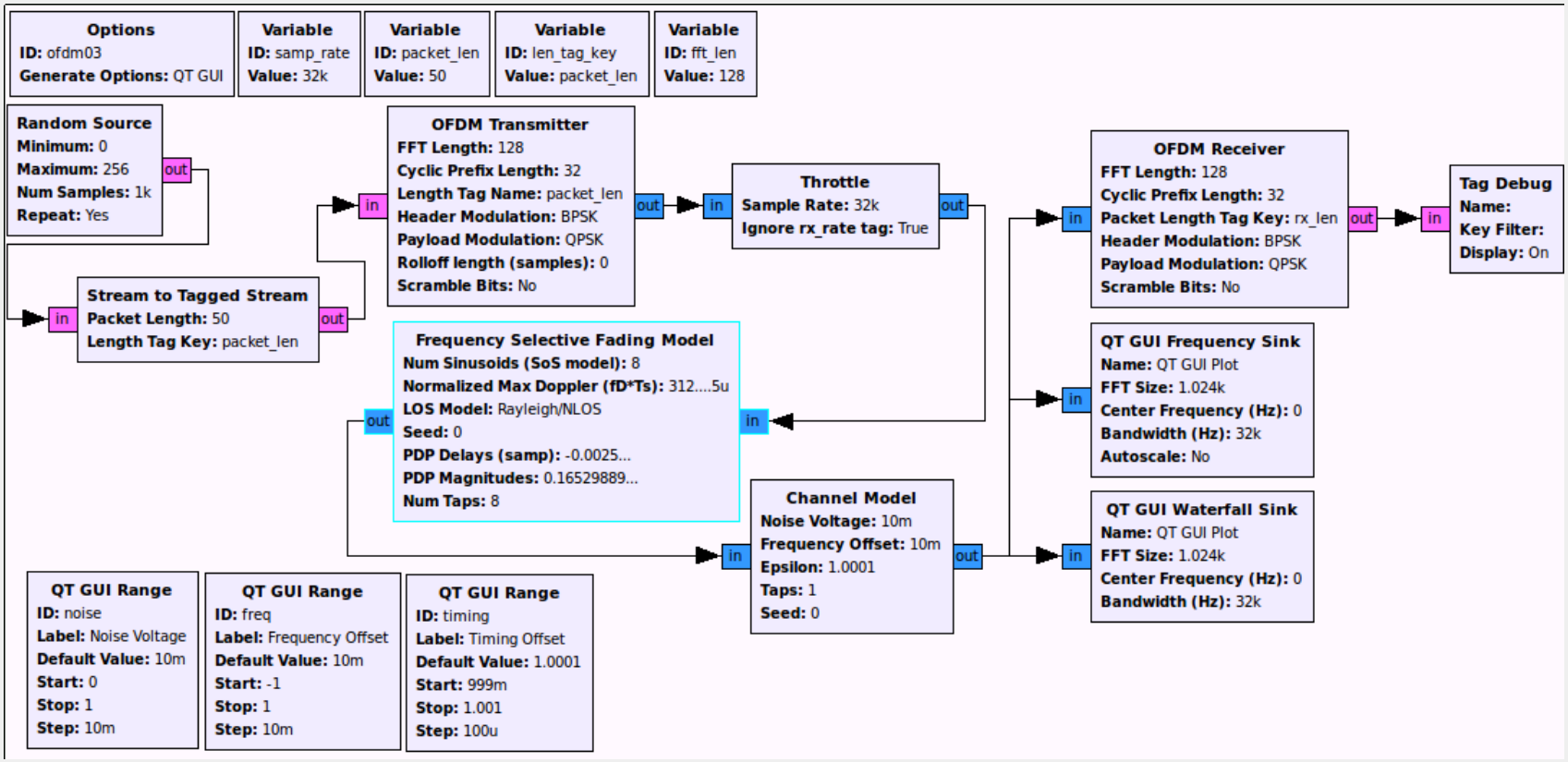
Add a simple, static channel



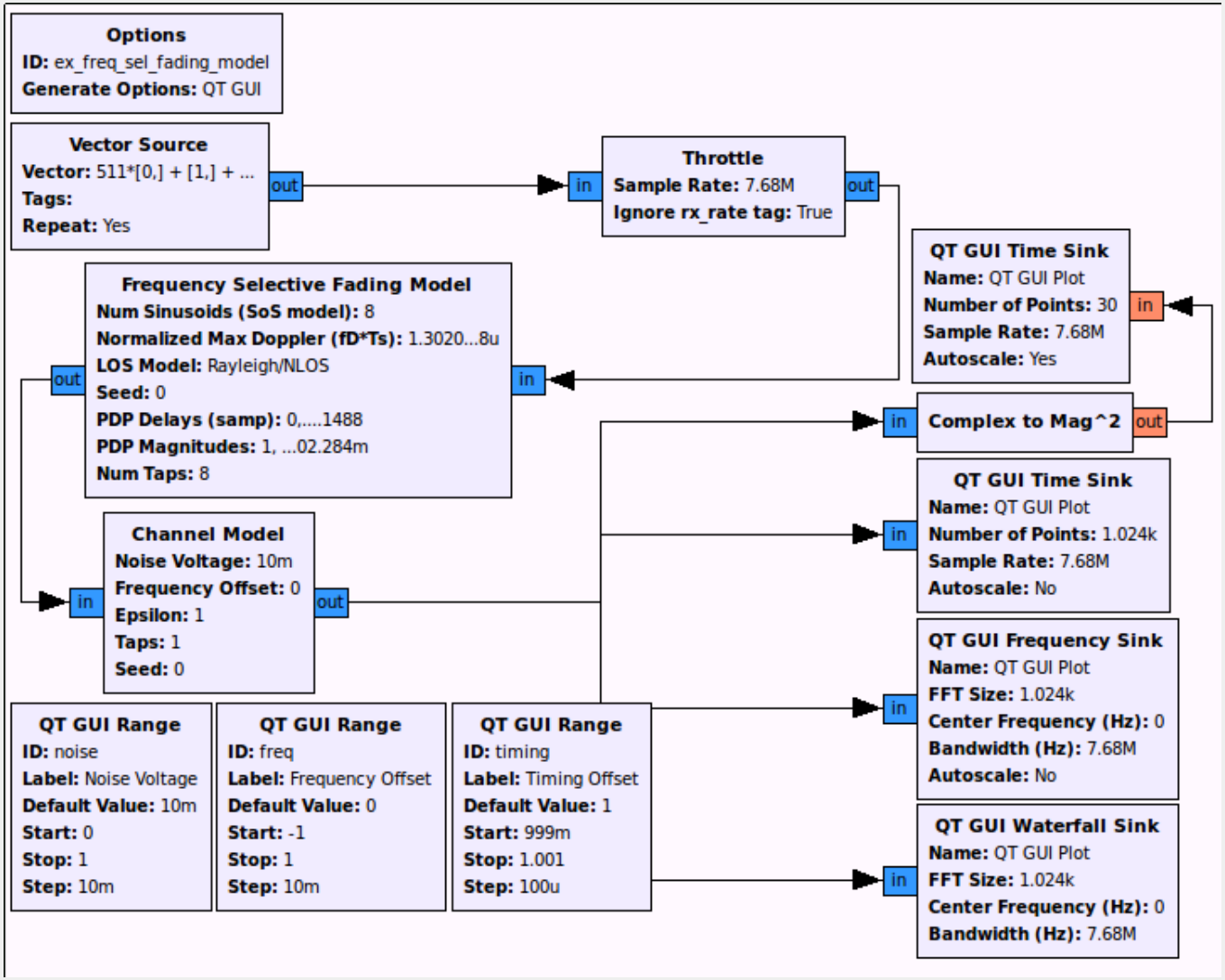
Basic Fading Model



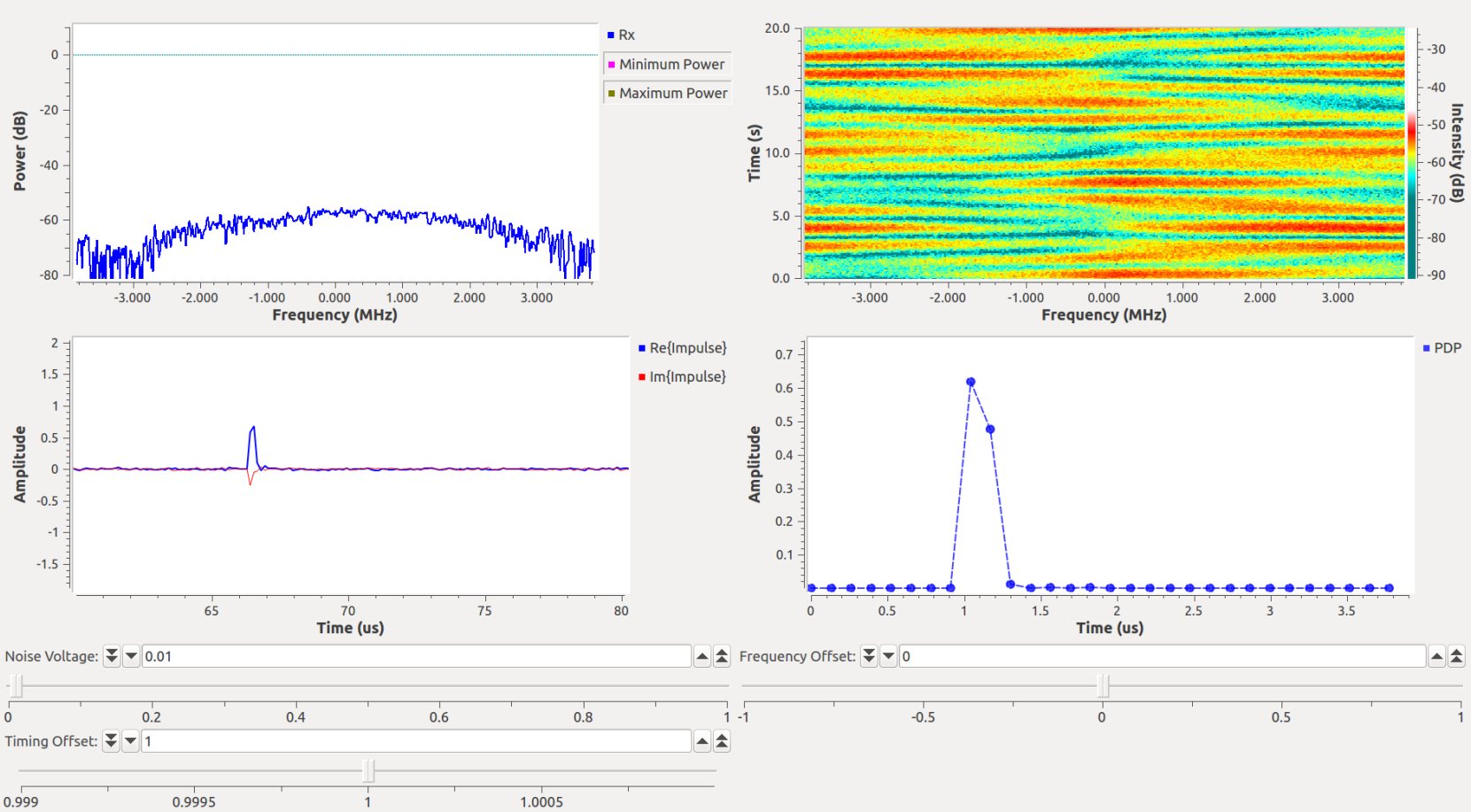
Fading Model with PDP



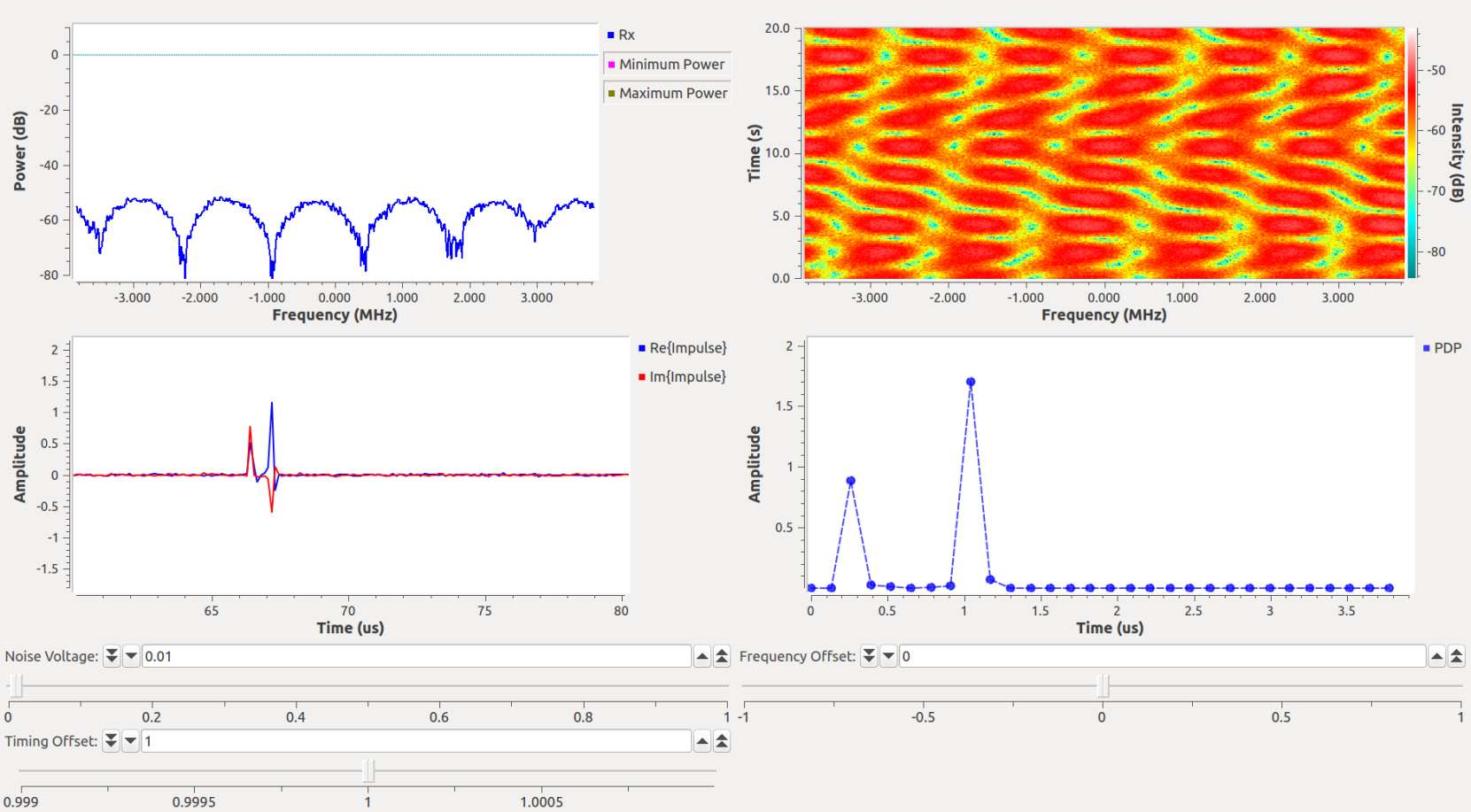
Experimenting with PDPs



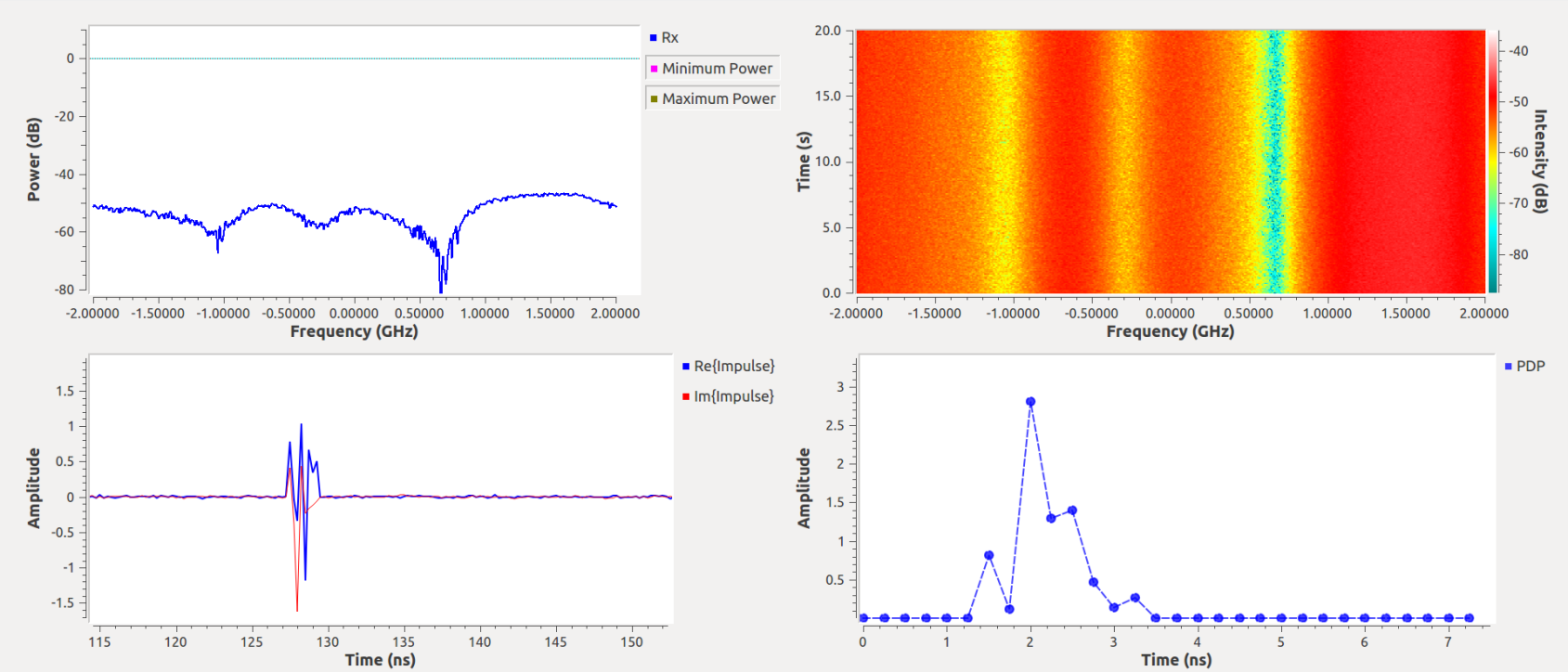
Sample Output: UMTS Pedestian A



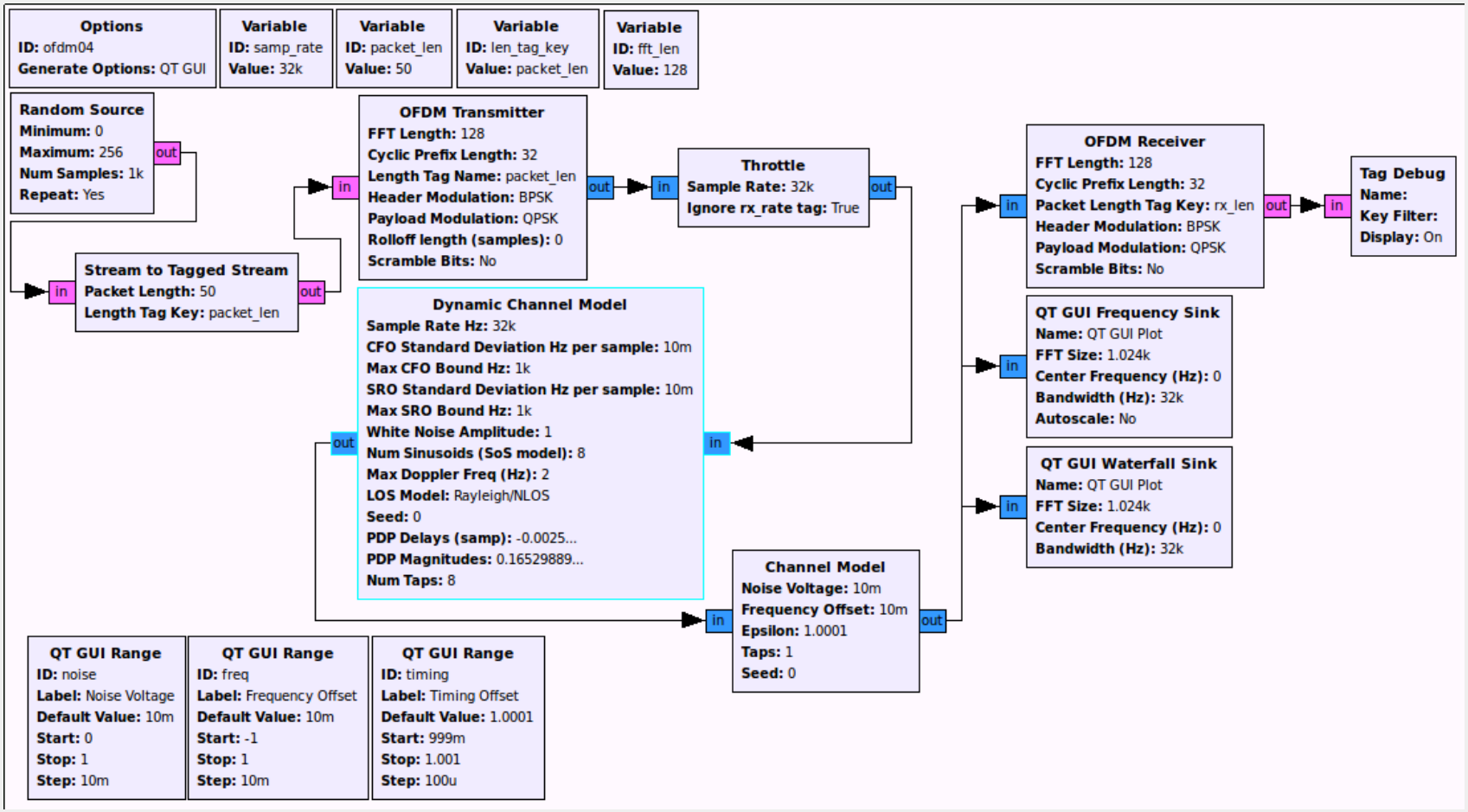
Sample Output: UMTS Pedestian B



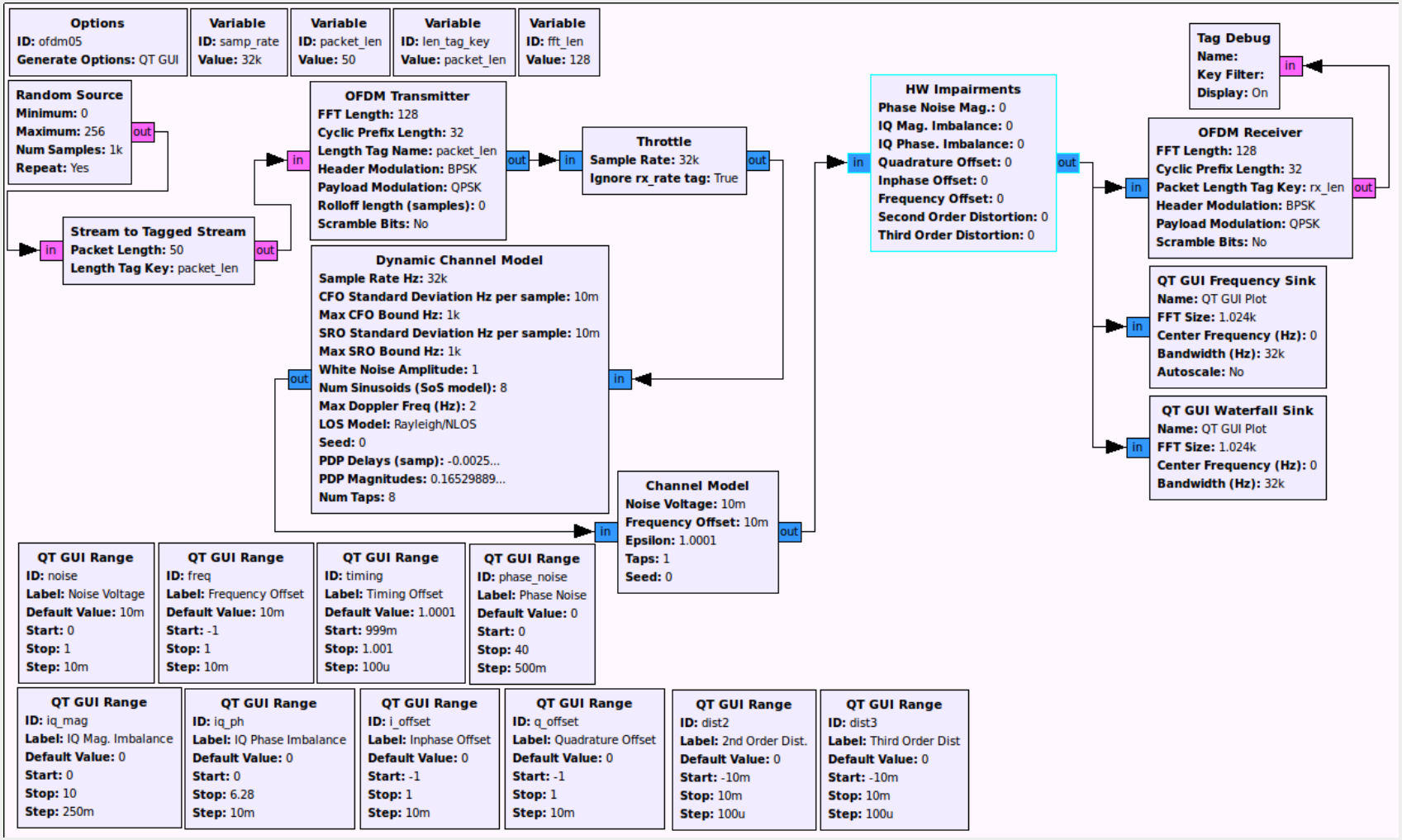
Sample Output: PDP Collected from Chris Anderson (USNA)



Fading Model, PDP, and Frequency Walk



Adding Hardware Impairments Model



Hardware Inherent

- Non-linearity
- Quantization
- Phase Noise
- Passband Shape and Group Delay

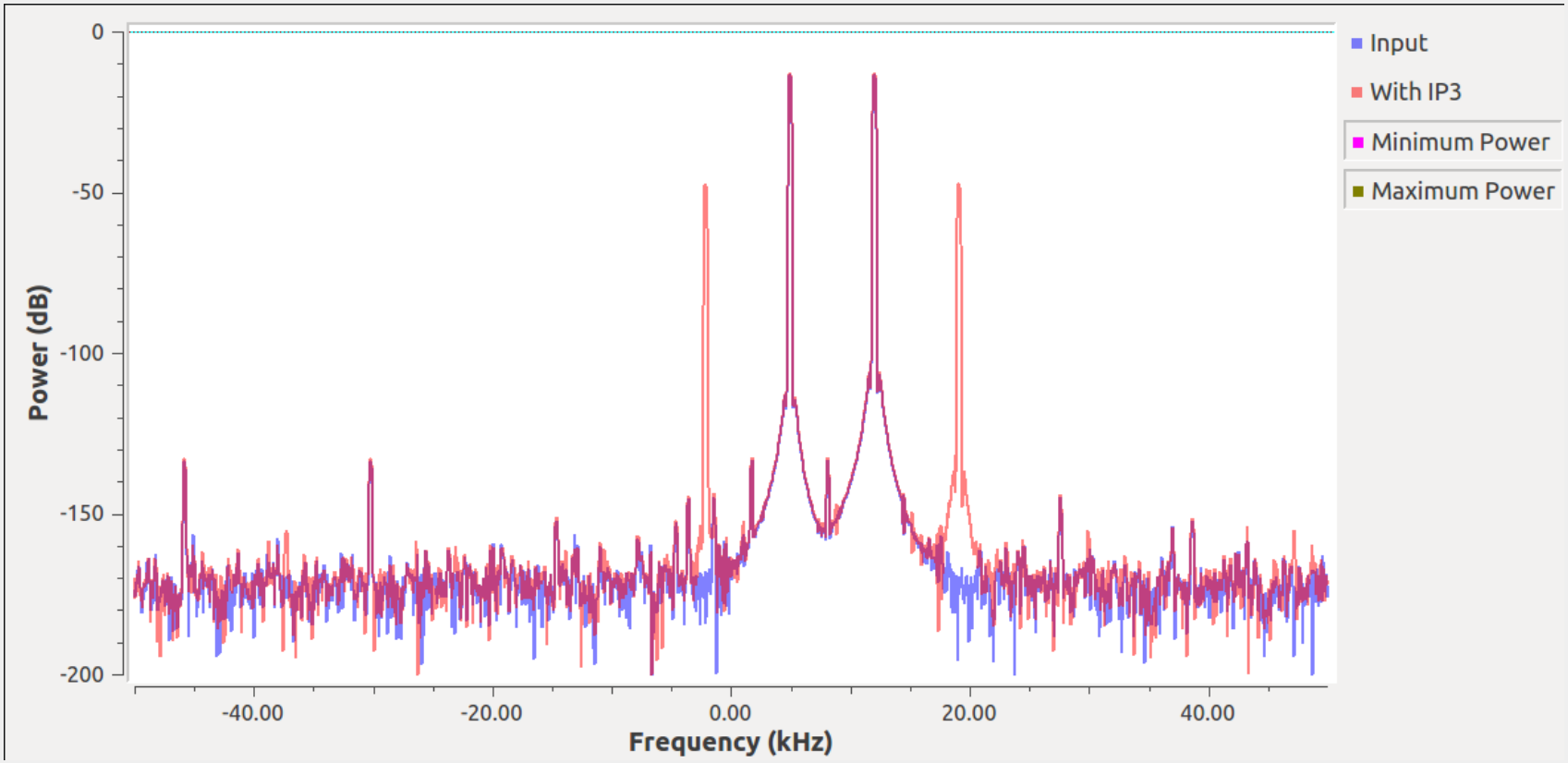
Non-linearity

- Output not a multiple of the input
  - Transfer function depends on amplitude
- Primary mechanism in semiconductor amps is clipping behavior as you approach maximum output
- $V_{out}=k_1 V_{in} + k_2 V_{in}^2 + k_3 V_{in}^3 + ...$
  - $cos^2 x = \frac{1+cos2x}{2}$
  - $cos^3 x = cos(3x) + 3cosxsin^2 x$
  - Output contains frequencies not in the input (harmonics and mixing products)
- Not just amplifiers (mixers, capacitors, inductors, even connectors)
- More complex models
  - Volterra Series
  - AM-AM and AM-PM

Third order non-linearity

- Third order the most important
- Typically modeled with third order intercept (IP3, IIP3, OIP3)
  - Intercept point is the [extrapolated] point at which intermod products would equal desired products
  - Typically ~10dB above P1dB
  - Don't actually operate at that point!
- $P_{IMD3} = 3P_{signal} - 2IP_3$
  - IMD3 products increase 3x as fast as input
  - IMD products appear at $2f_1 \pm f_2$, $2f_2 \pm f_1$, $3f_1$, $3f_2$

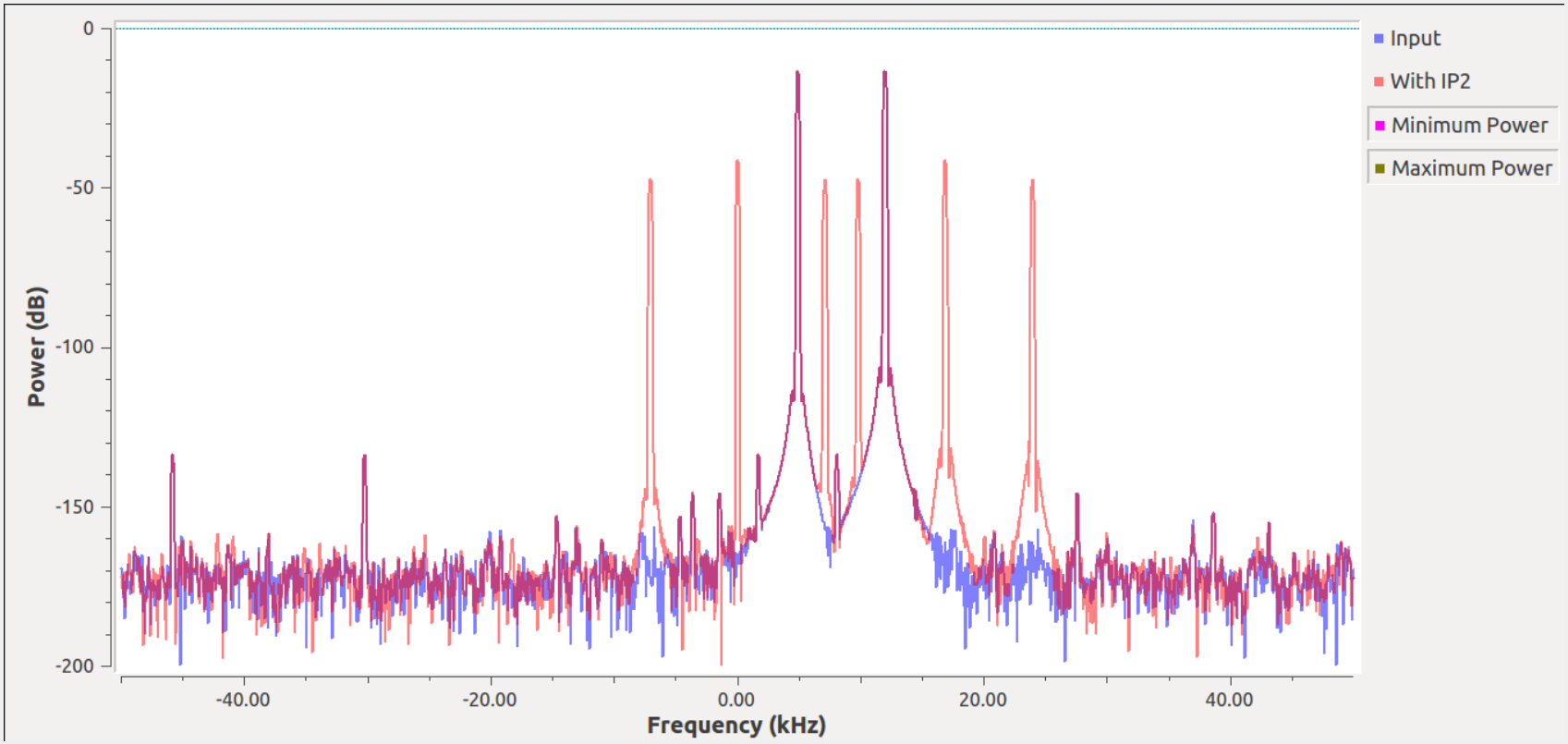
IP3 of Non-Harmonically Related Signals



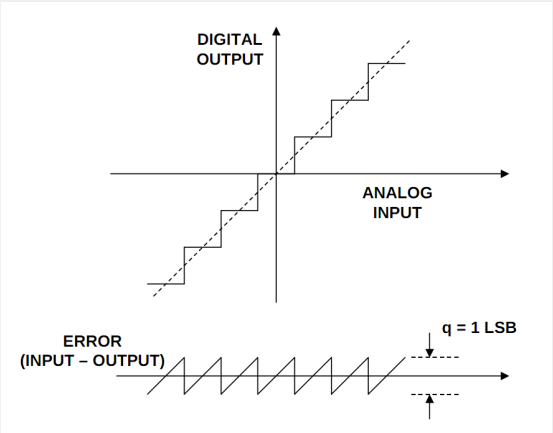
Second order non-linearity

- 2nd order products fall at DC, $f_1 \pm f_2, 2f_1, 2f_2$
- DC IMD2 product often mistaken for DC offset
- Only a problem in certain situations
  - Band of interest is greater than 1 octave
  - Band of interest includes DC
  - Direct Conversion receivers

IP2 of Non-Harmonically Related Signals



Quantization

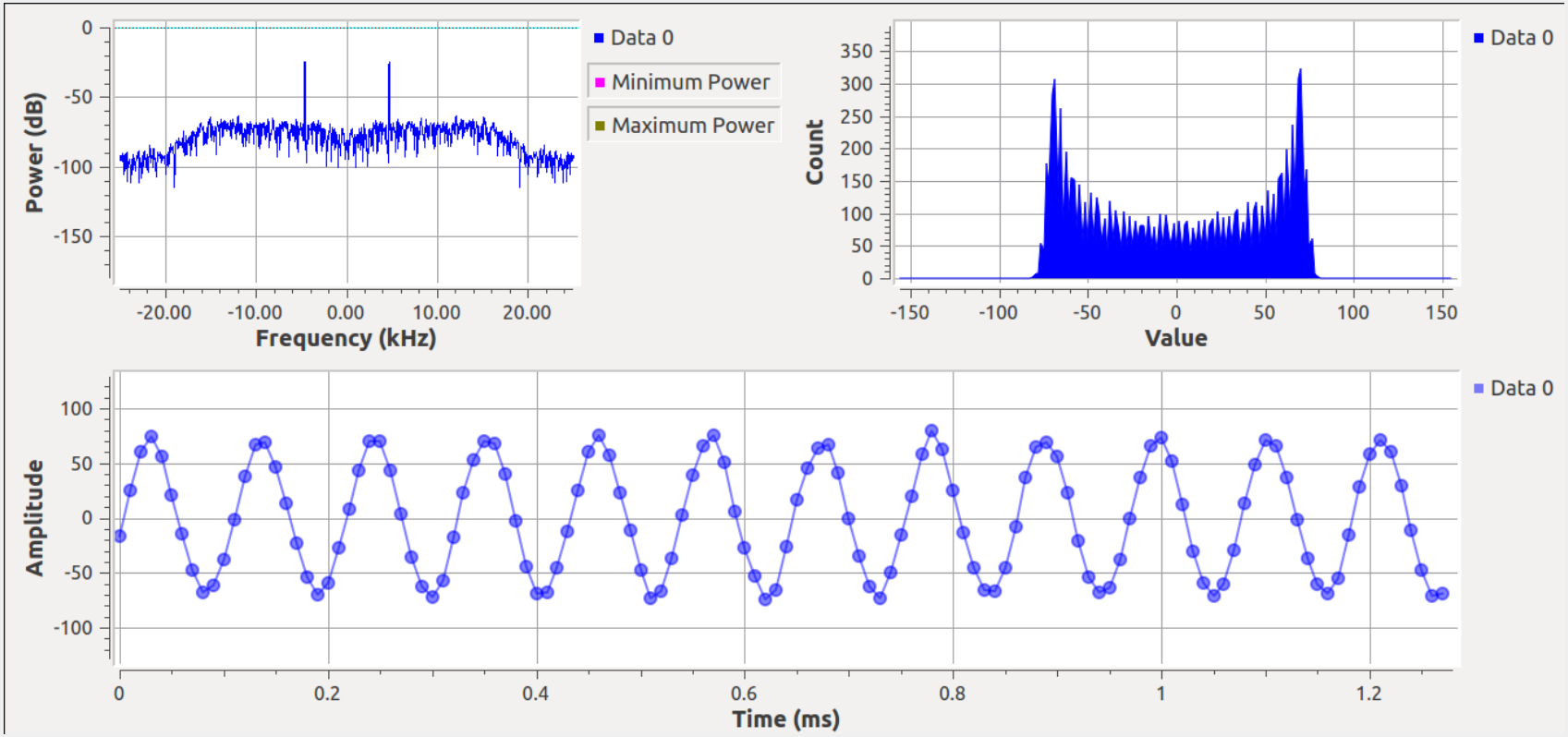


- Not to be confused with discretization (i.e. time steps)
- Inherent in digital systems
  - Finite bit widths in ADC, DAC
  - Costs of digital processing, storage, transmission
    - Cost of a Multiply operation is proportional to bits^2
  - Even floating point numbers are quantized

Quantization, cont'd

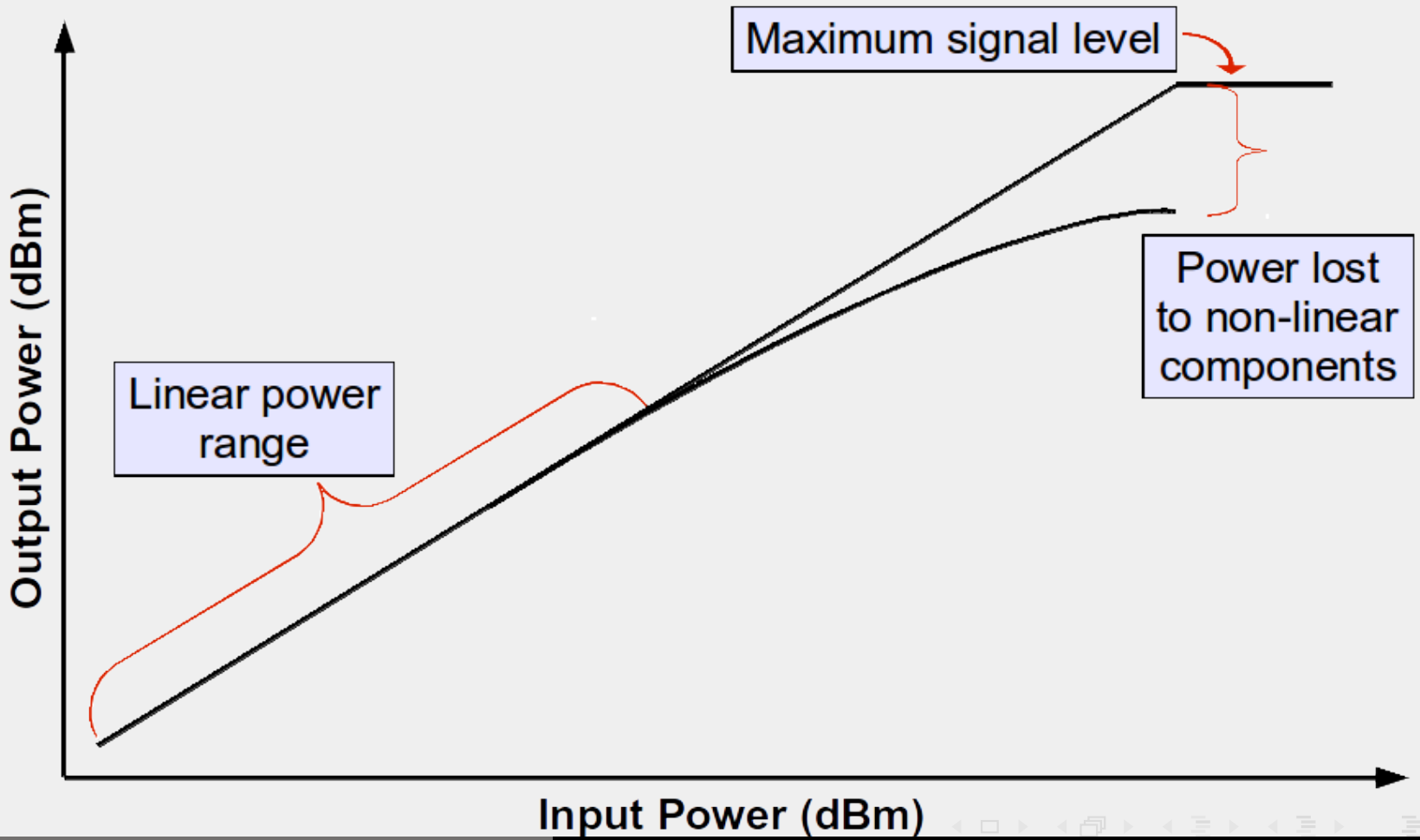
- Quantization results in noise
  - Often modeled as AWGN
    - Beware of correlated quantization noise ($f/f_s \backsimeq M/N$)
  - $SNR = 6.02N + 1.76dB$
- Non-ideal ADC/DAC behavior causes similar problems to correlated noise

Quantization Demo

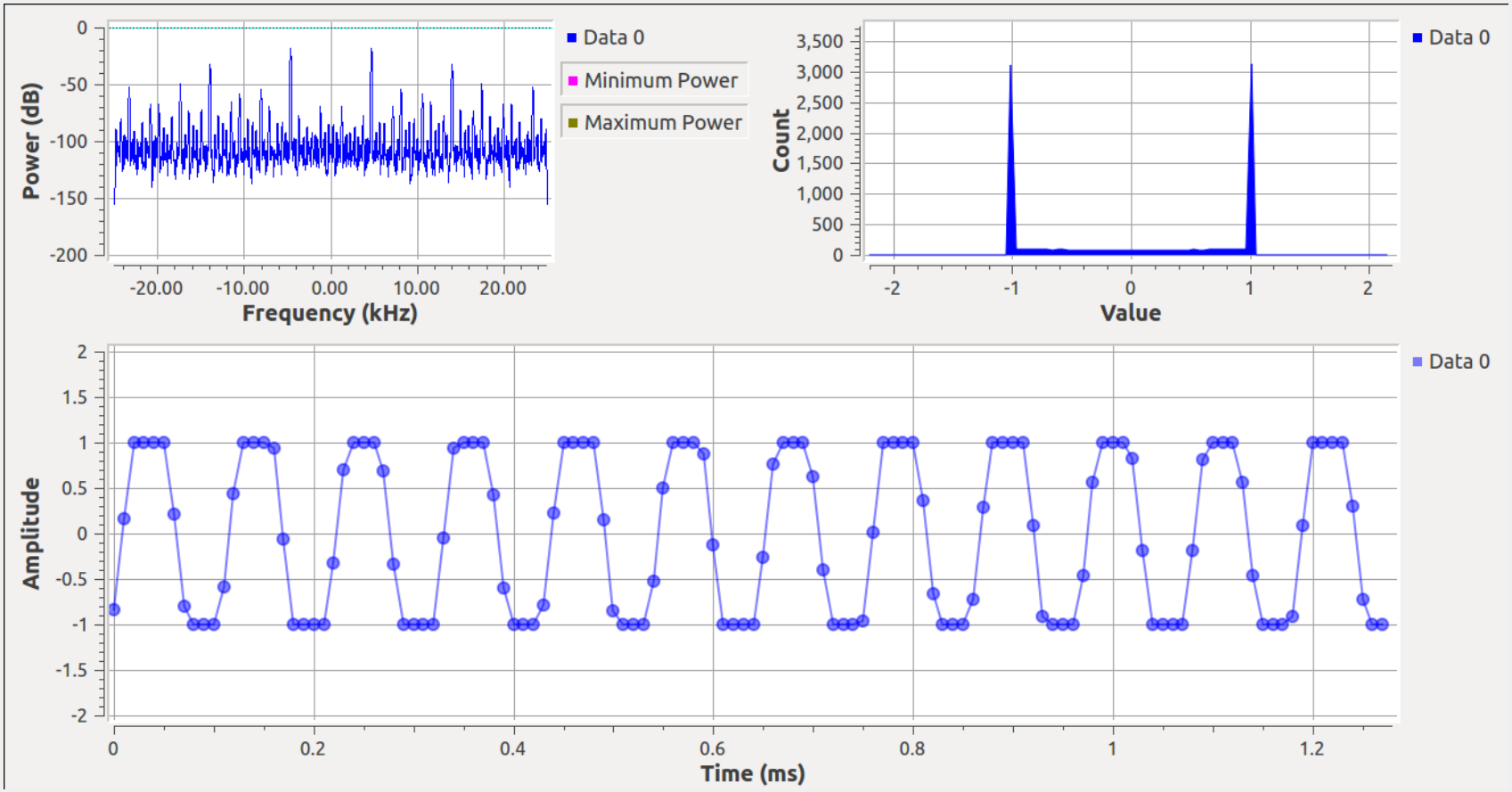


Clipping

- ADCs (and/or other components) have a maximum input range

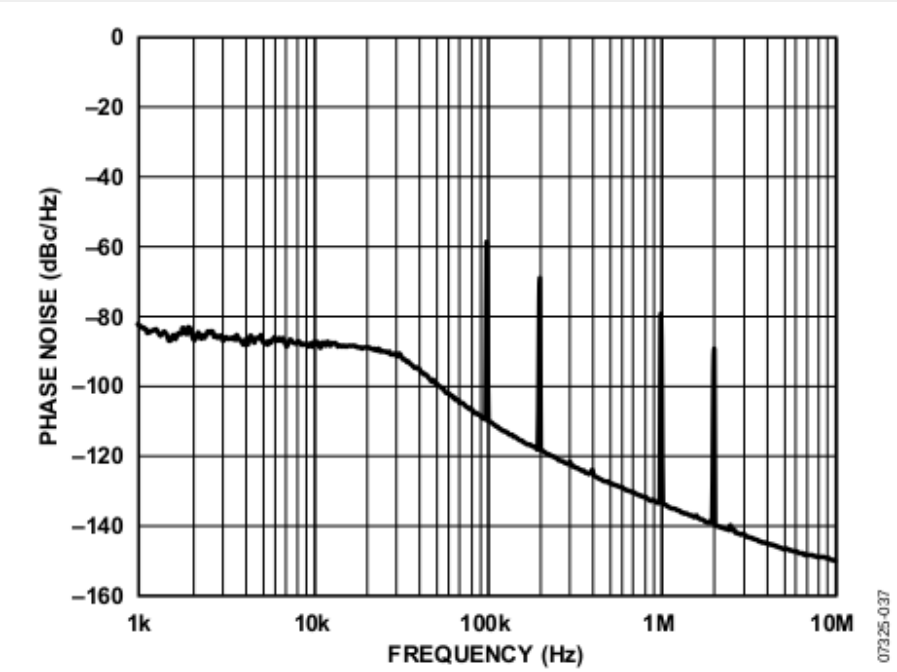


Clipping Demo



Phase Noise

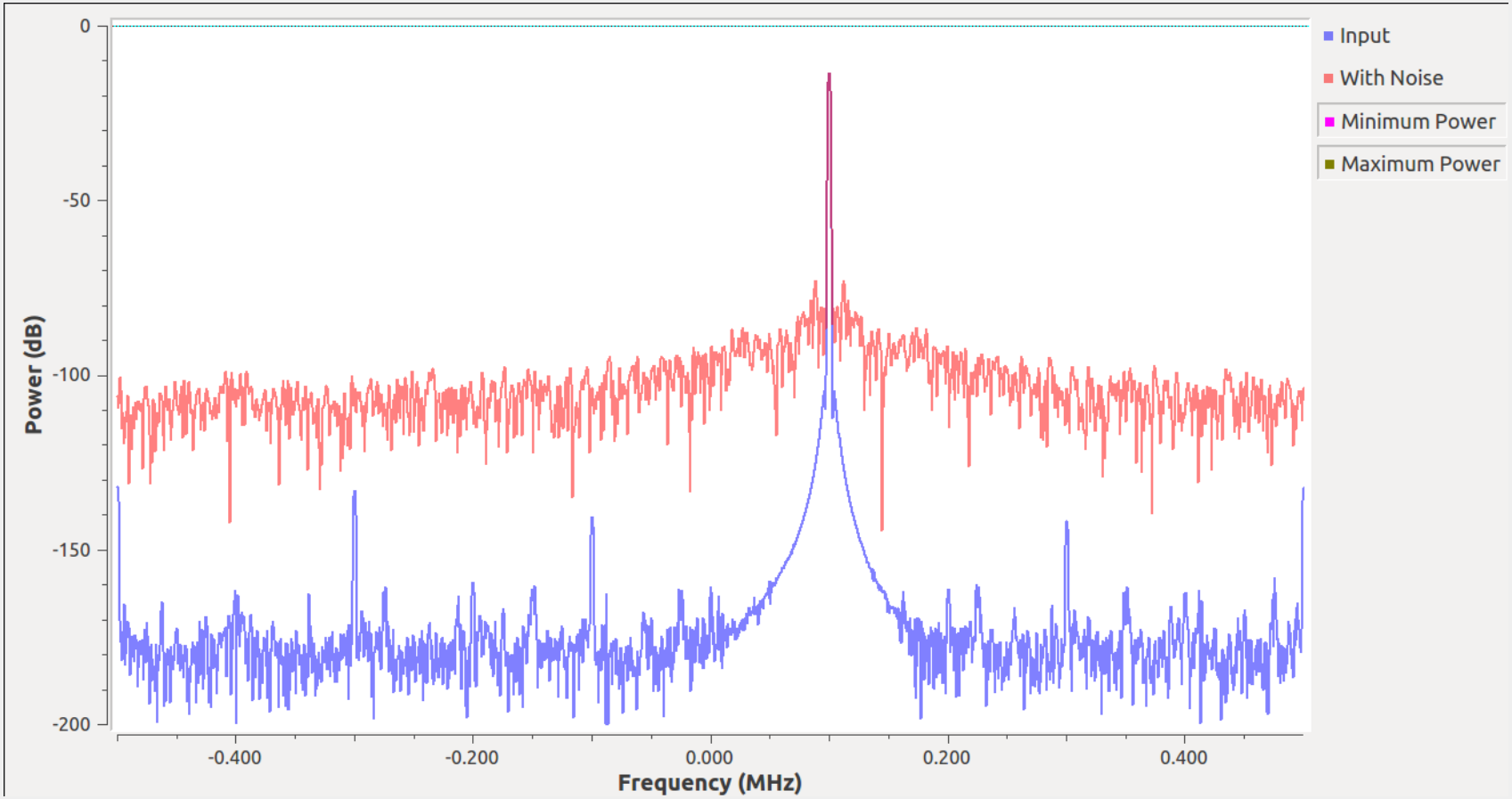
- Random phase perturbations on an oscillator
- Specified as dBc/Hz at an offset from carrier
  - i.e. -100dBc/Hz at 100kHz offset
- Modeled by the Leeson phase noise equation
- Spurs are a related phenomenon with similar symptoms



Phase Noise, cont'd

- Always causes self noise
  - increasing signal doesn't help
- -100dBc/Hz doesn't sound like much
  - Over a 10 MHz BW signal that equates to -30dBc
    - No QAM 256 for you!
- Total integrated phase noise often specified
  - I.e. 1.5 degrees RMS in a 20kHz to 80 MHz BW
- On TX causes adjacent channel emissions, broadband noise floor
- On RX mixes strong adjacent signals onto desired signal

Phase Noise Simulation



Architecture Specific

- DC Offset
- IQ Balance

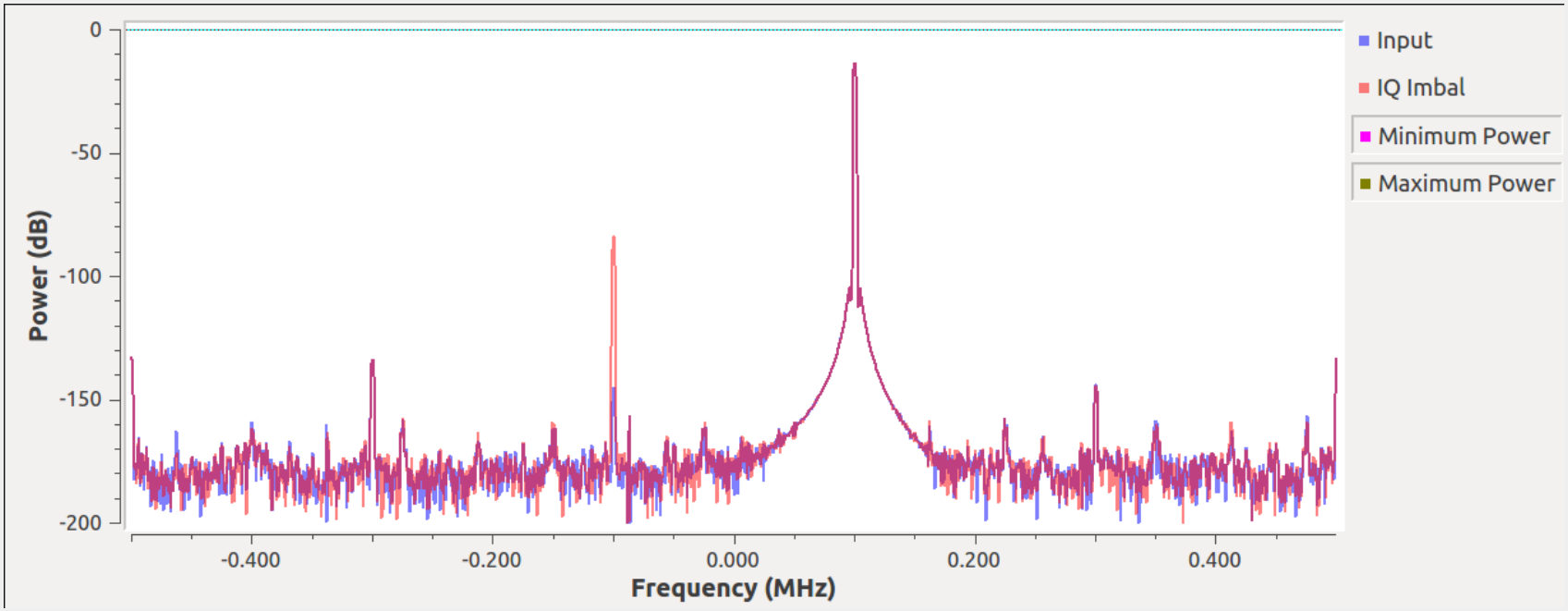
DC Offset

- Causes
  - Component mismatch
  - LO leakage
  - 2nd order distortion
- Varies with time, temp, frequency, voltage, moon phase, etc.
- Produces self interference
- Remedies
  - Ignore DC (use low-IF or ignore DC bin in OFDM)
  - AC-couple
  - Highpass filter (receiver only)
  - Estimate and subtract in either analog or digital domains
    - must be done at true baseband
    - much easier on the receiver

IQ Imbalance

- Magnitude imbalance caused by gain mismatch between paths
- Phase imbalance caused by
  - imperfect 90 degree phase shift in LO
  - mismatched phase or group delay between I and Q paths
- Varies with time, temp, frequency, voltage, moon phase, etc.
- Effects
  - Self interference
  - Out of channel leakage on transmit
  - Susceptibility to out of channel interference on receive
  - Inherently non-LTI since it generates new frequencies

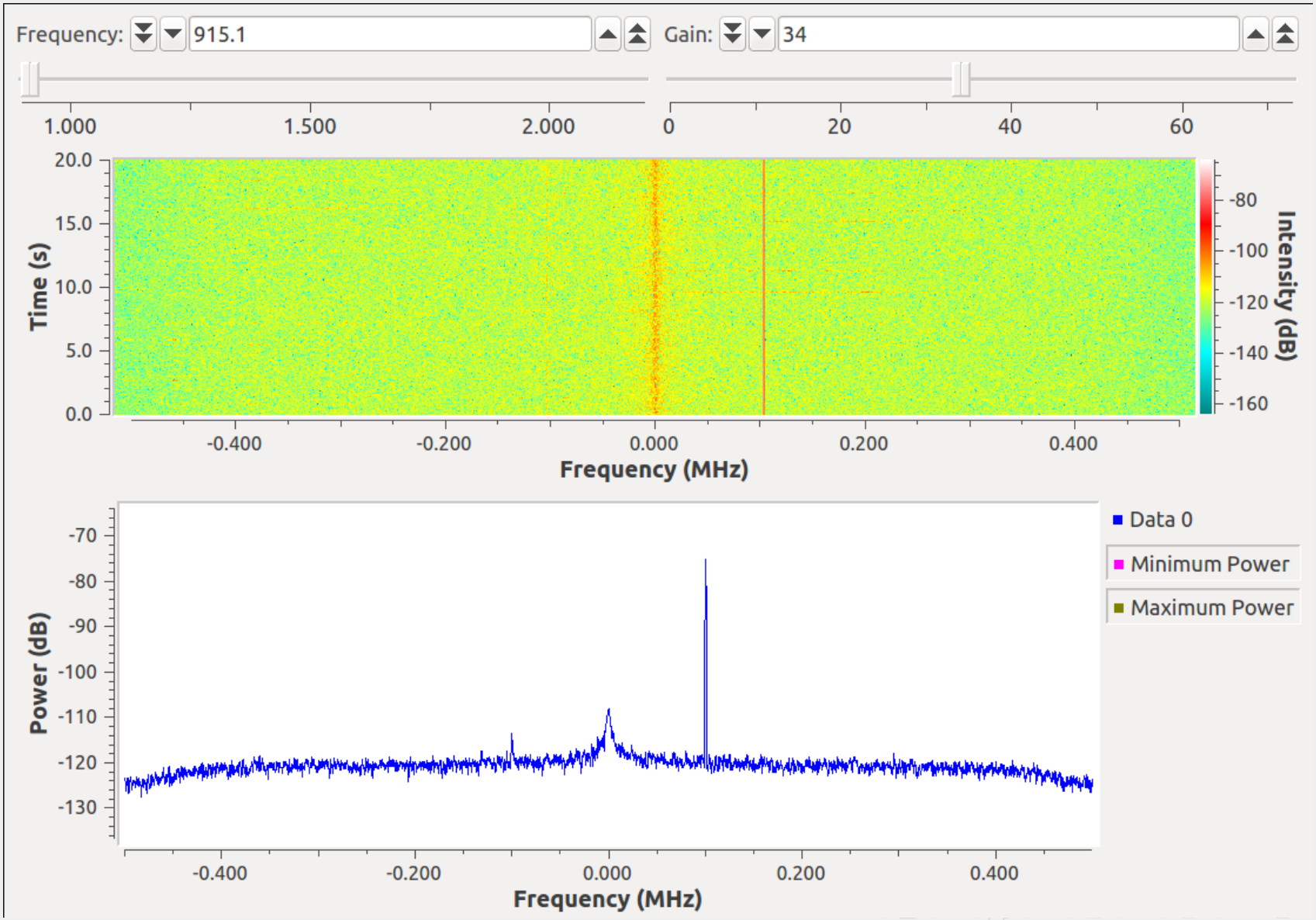
IQ Imbalance Simulation



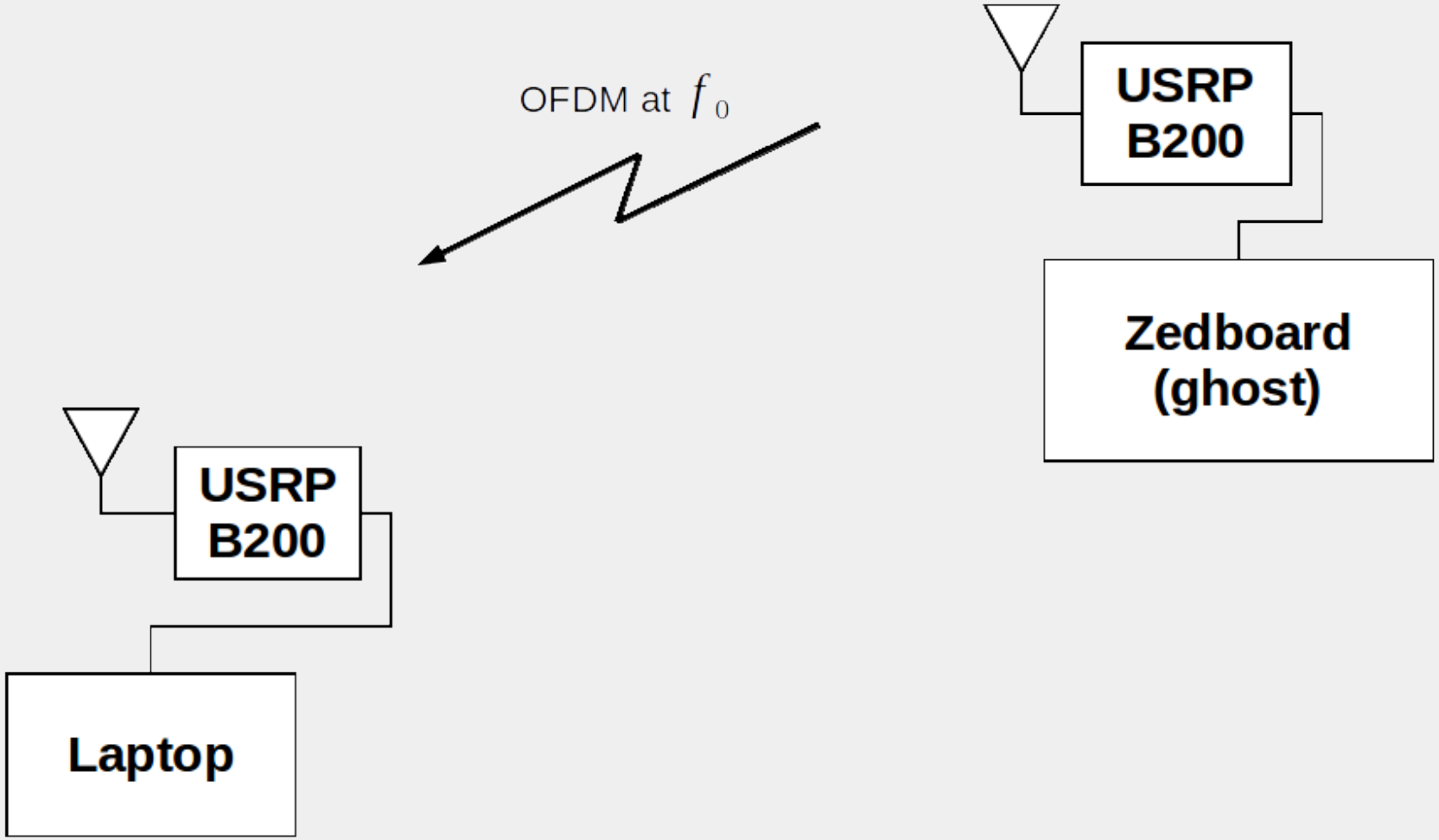
Fixing IQ Imbalance

- Remedy is
  - estimate the relative I and Q magnitude error and scale appropriately
  - estimate the relative phase and rotate components appropriately
  - Must be done at true baseband
  - Much easier on the receiver
- May be baseband frequency selective
  - Must scale magnitude and phase differently for different frequencies
    - Requires multi-tap filter

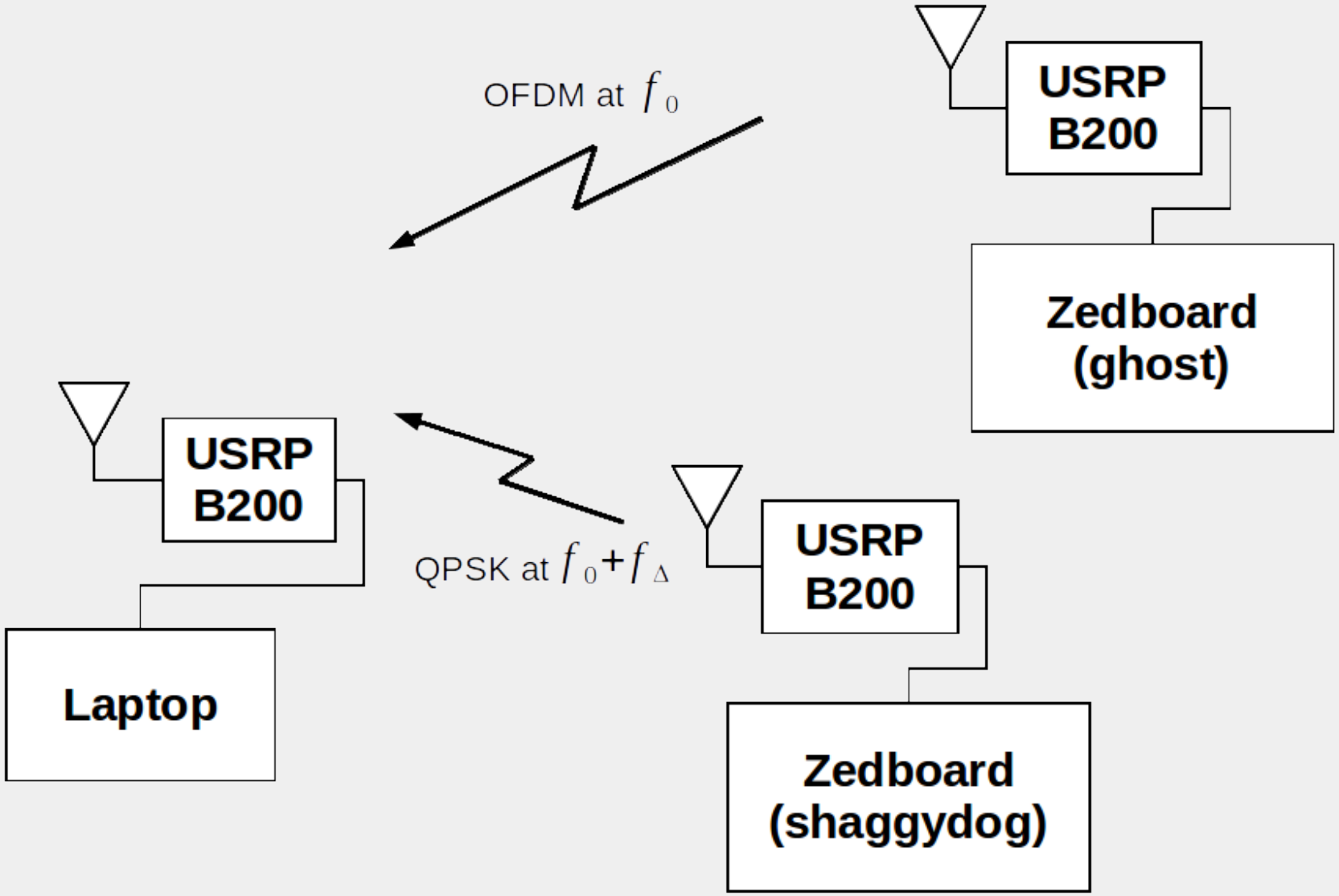
DC Offset and IQ Imbalance in Action



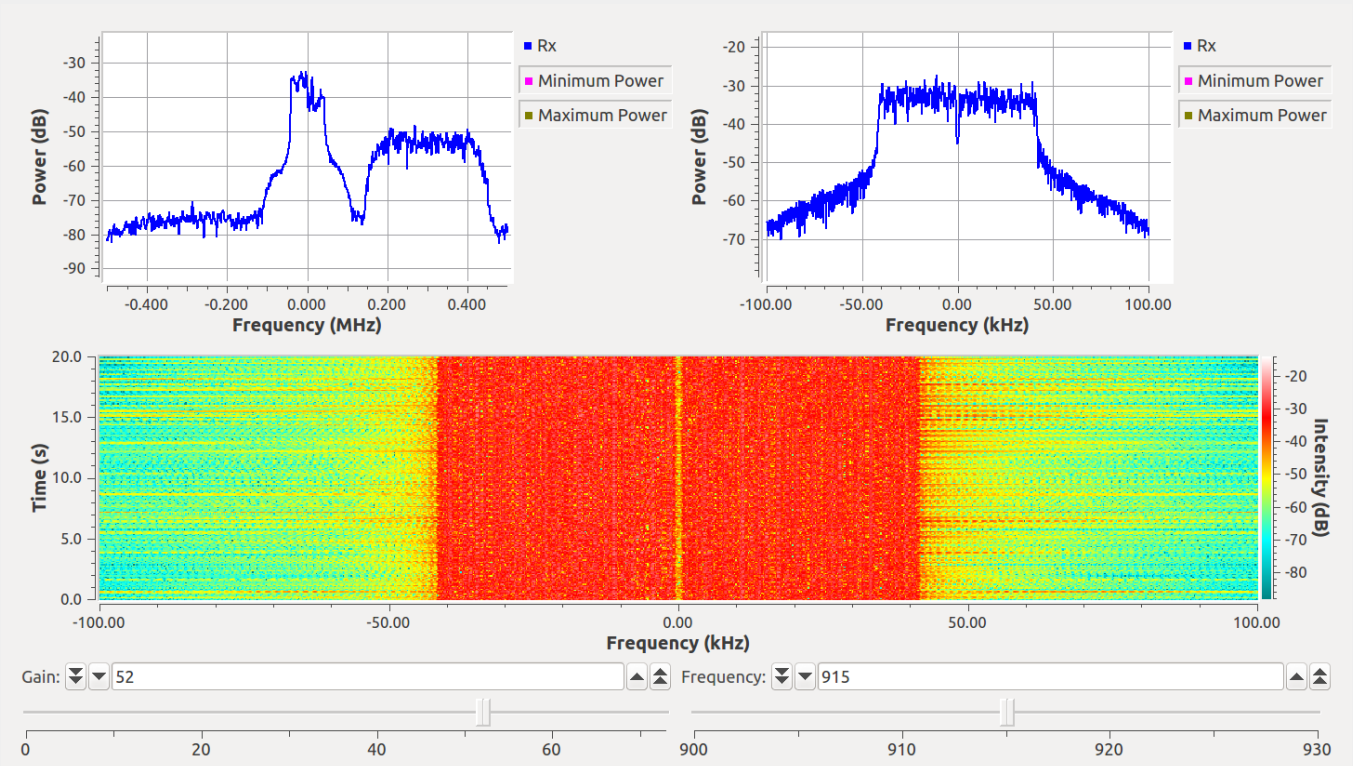
Simple Tx-Rx



Opportunistic Access

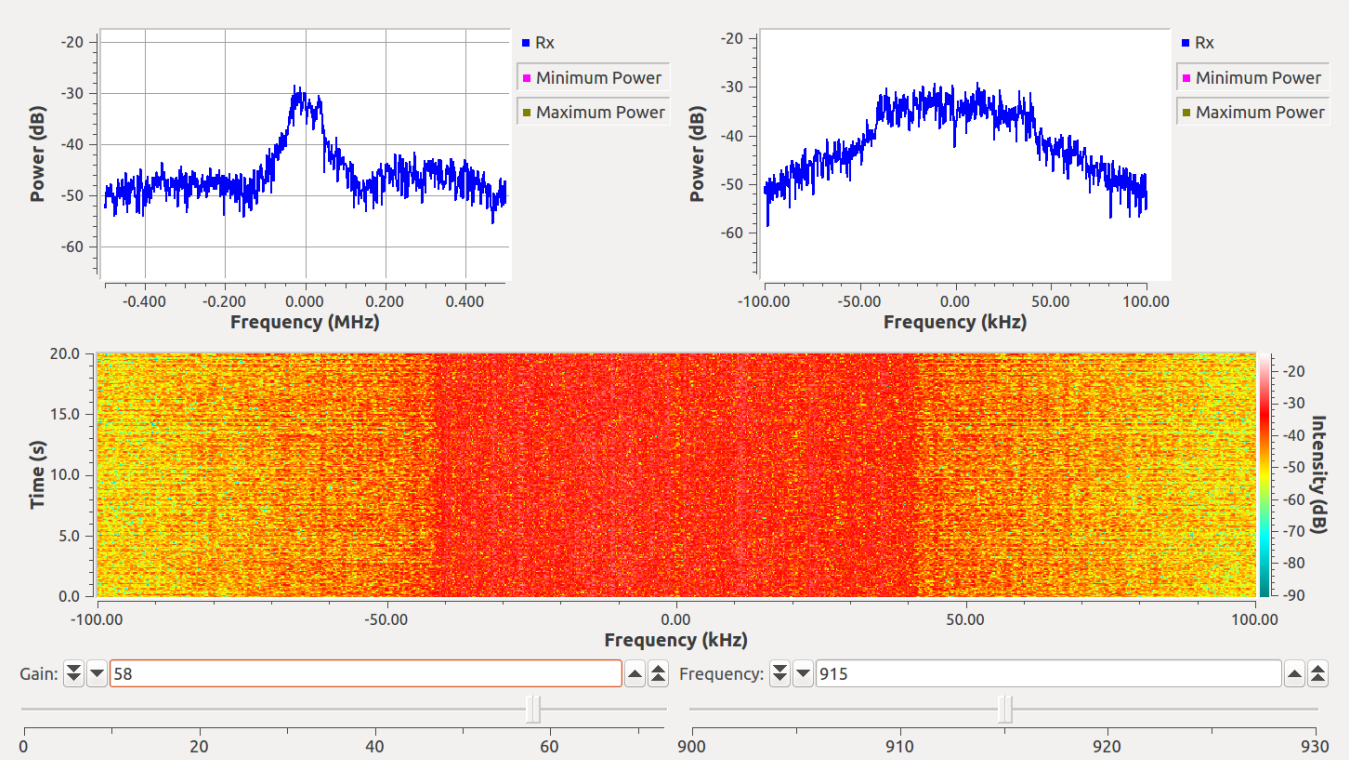


Sharing Spectrum



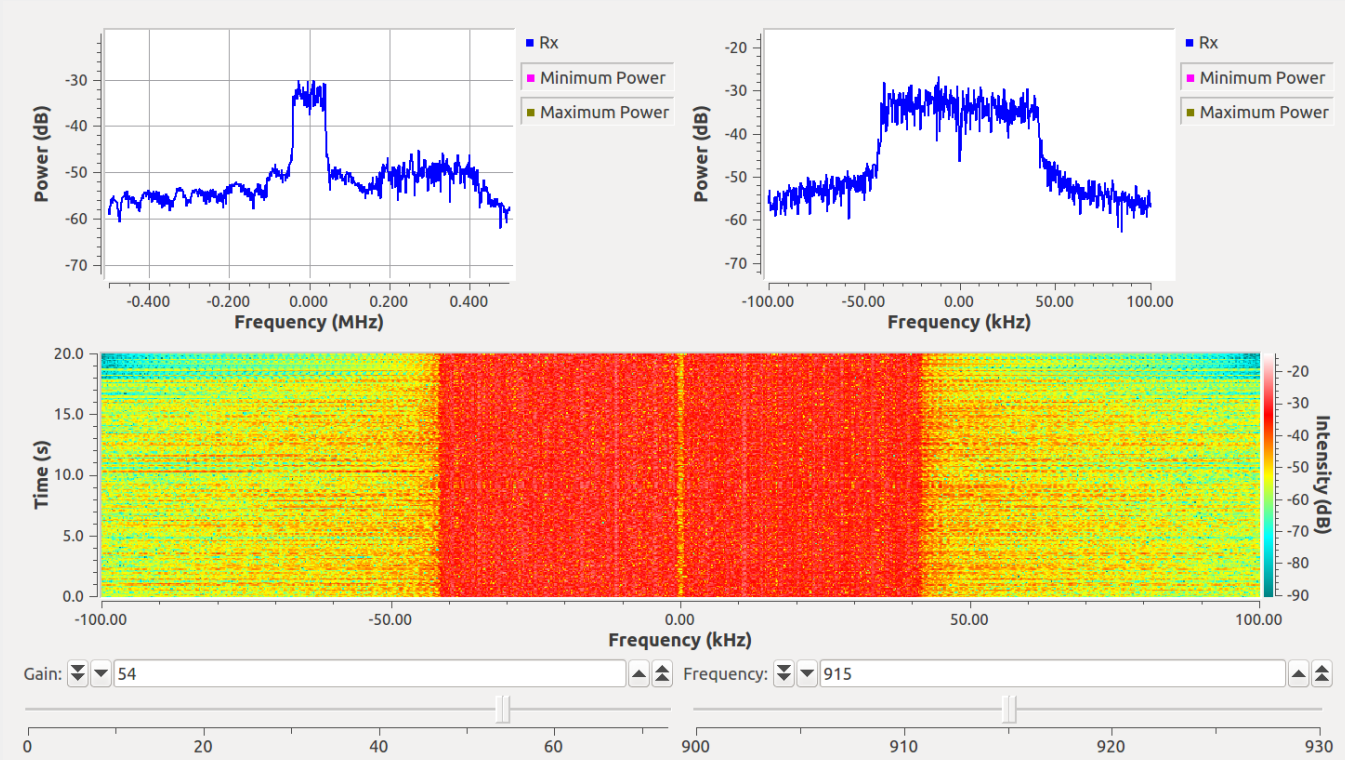
- Transmit gain of QPSK: 40 dB
- Receiver gain: 52 dB
- Packet Loss: 0%
- No observable distortion in received signal

What happened?



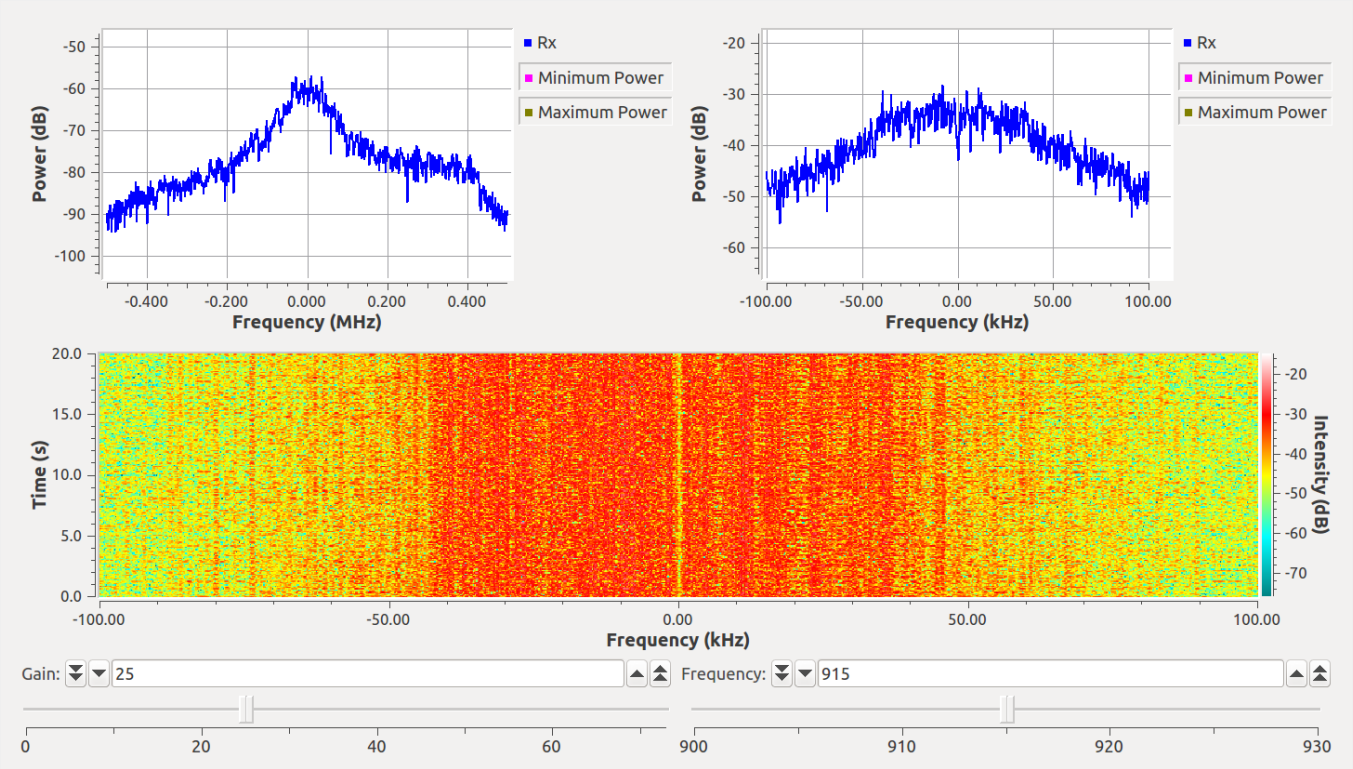
- Transmit gain of QPSK: 40 dB
- Receiver gain: 58 dB
- Packet Loss: just under 100%
- Obvious distortion in received signal

Even with just a few dB more gain



- Transmit gain of QPSK: 40 dB
- Receive gain: 54 dB
- Packet Loss: ~50%

Bad Tx Power Control



- Transmit power to RF too high causing distortion from the Tx

Filtering Tools in GNU Radio

"The need for filters intrudes on any thought experiment about the wonders of abundant information."

- Jame Gleik, *The Information*

To Cover

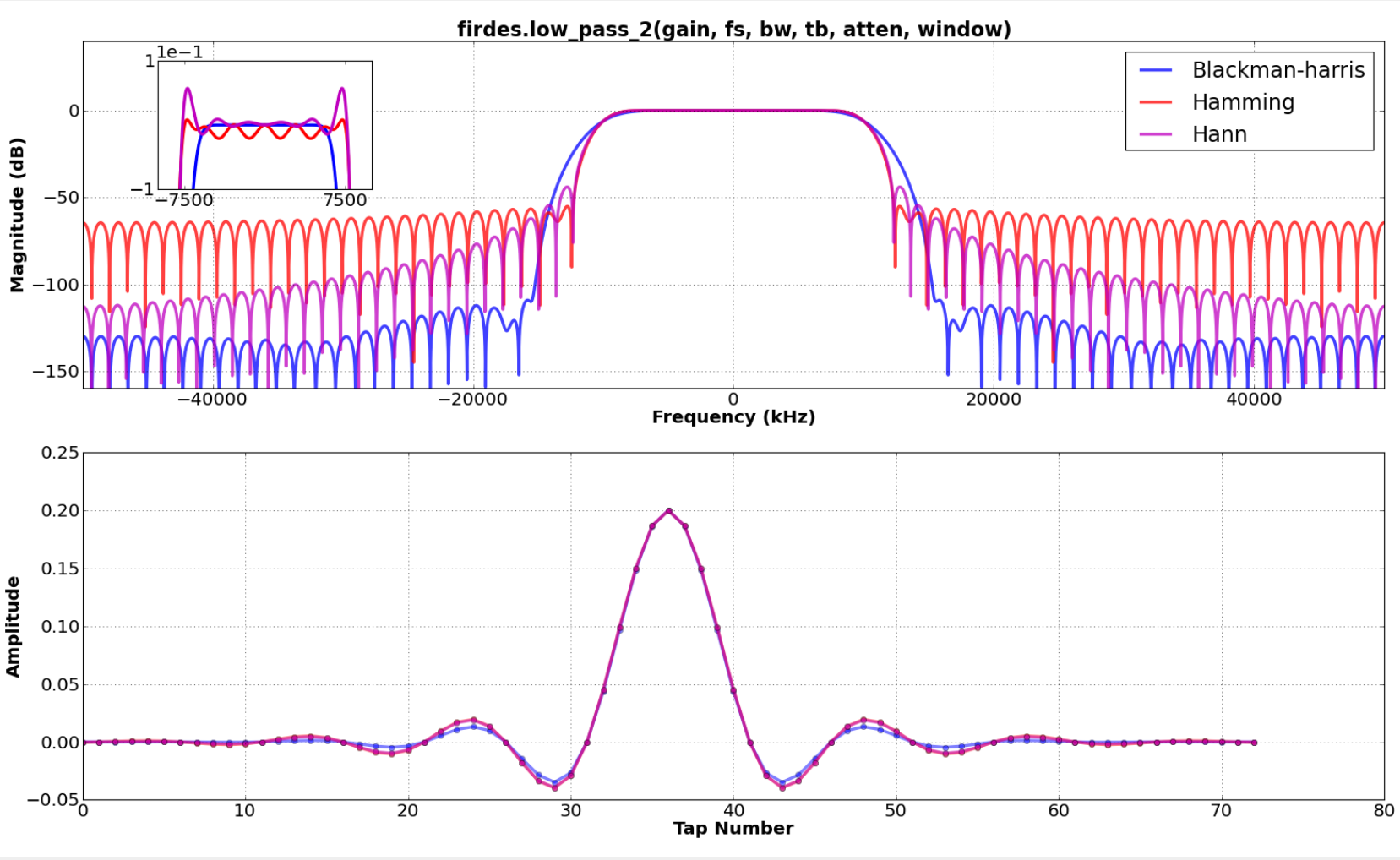
- Tools for designing filters
  - **firdes:** Builds standard filters using windows
  - **optfir:** Builds standard filters using Parks-McClellen
  - **gr_filter_design:** FIR and IIR design program
- Using filters
  - FIR: both convolution and fast convolution
  - IIR: limited
  - Polyphase filterbanks
- Using channelizers and synthesizers
  - How to build prototype filter
  - Channelizing example
- Reconstruction filters
  - How to build the prototype filter
  - Using an impulse response to verify

Filter Design Tool: firdes

- Design standard FIR filters using windowing method.
- Supports:
  - Low pass
  - High pass
  - Band pass (and complex band pass)
  - Band reject (i.e., band notch)
- Specialty filters:
  - Root raised cosine
  - Gaussian
  - Hilbert

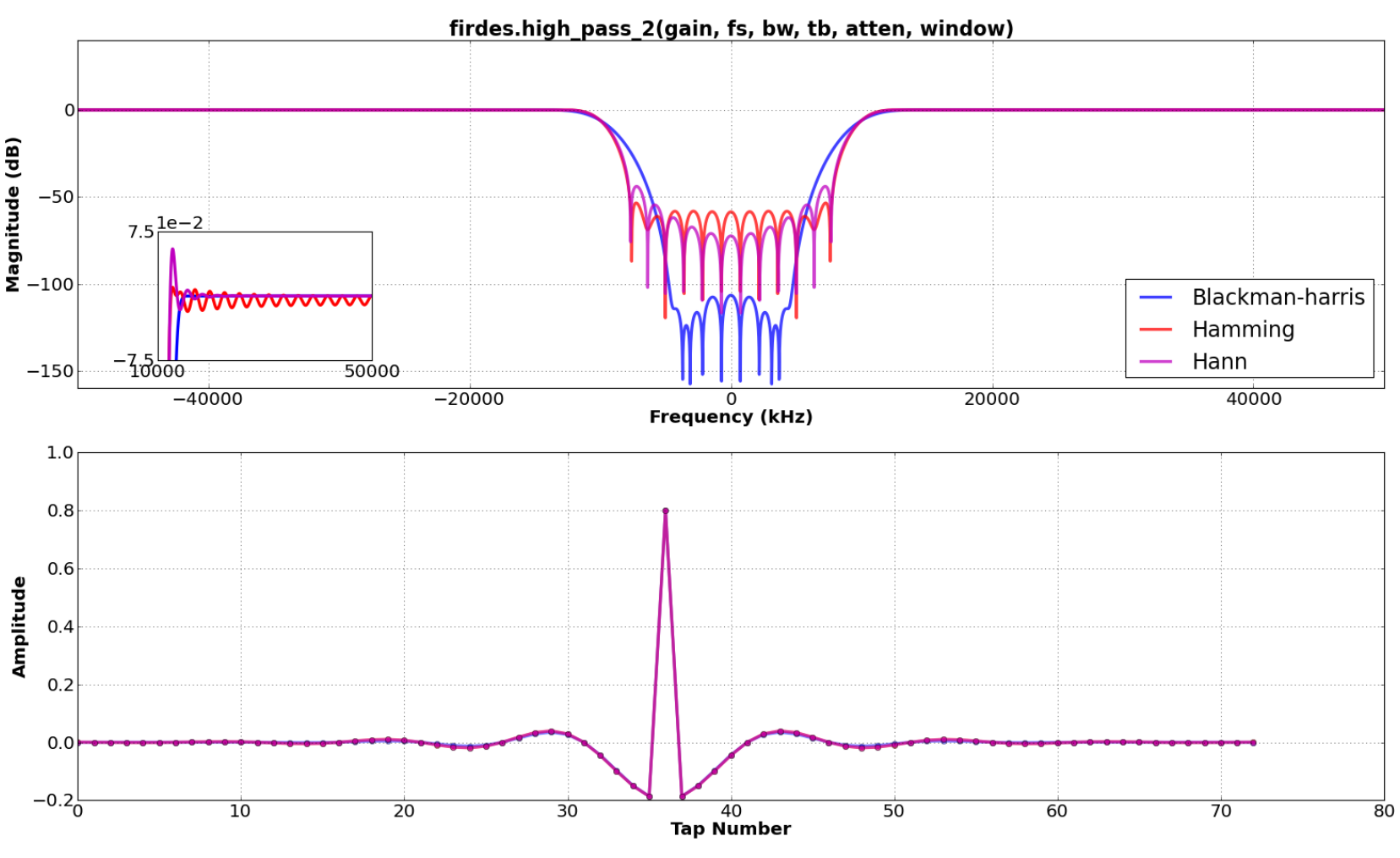
firdes designed filters

Low Pass Filter



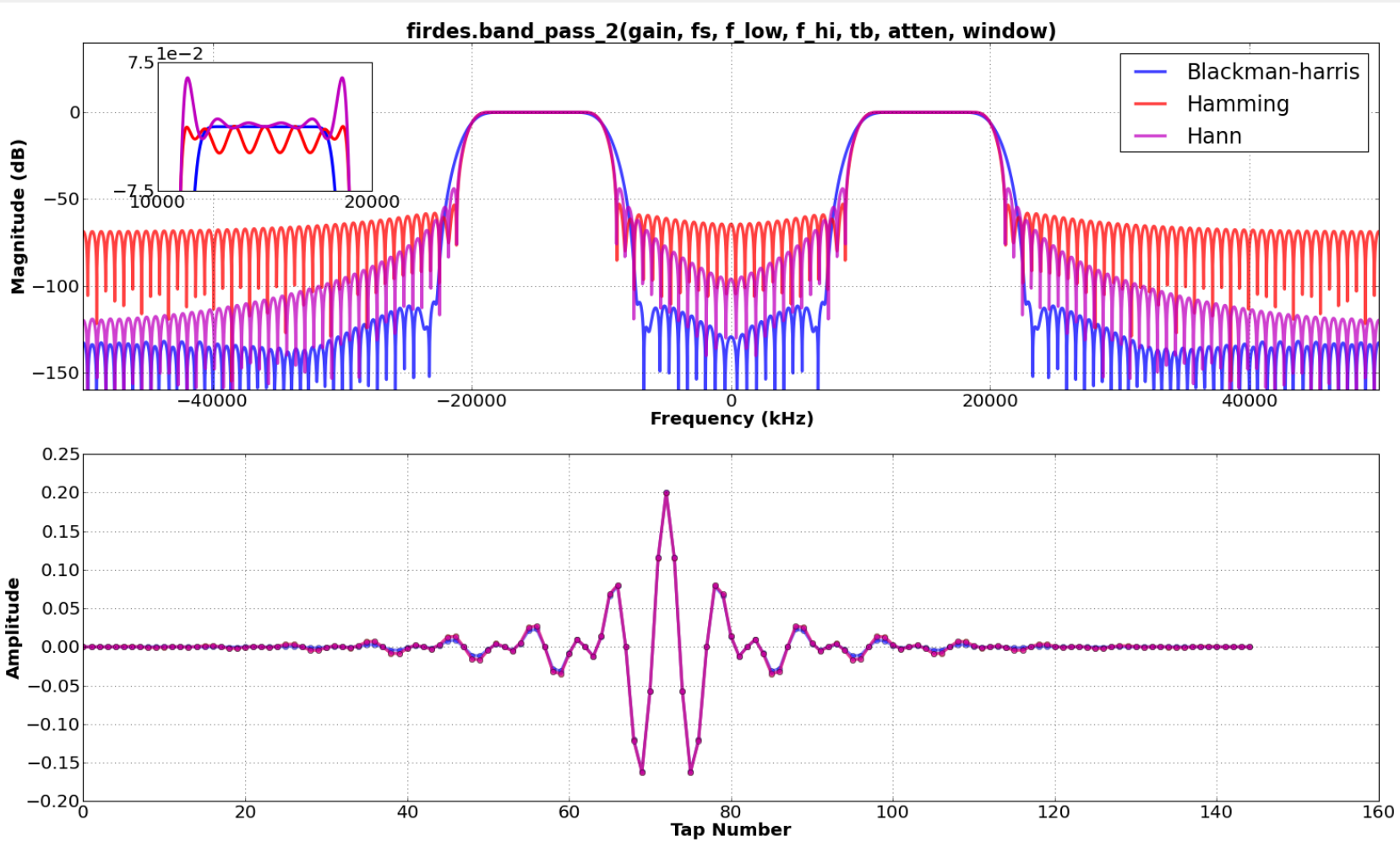
firdes designed filters

High Pass Filter



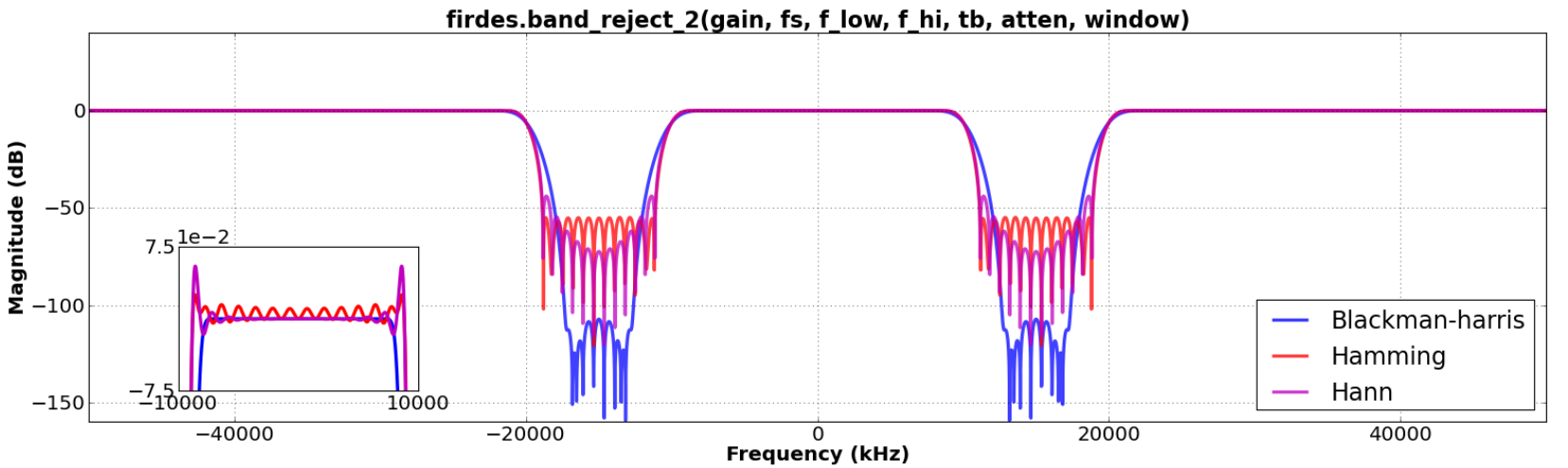
firdes designed filters

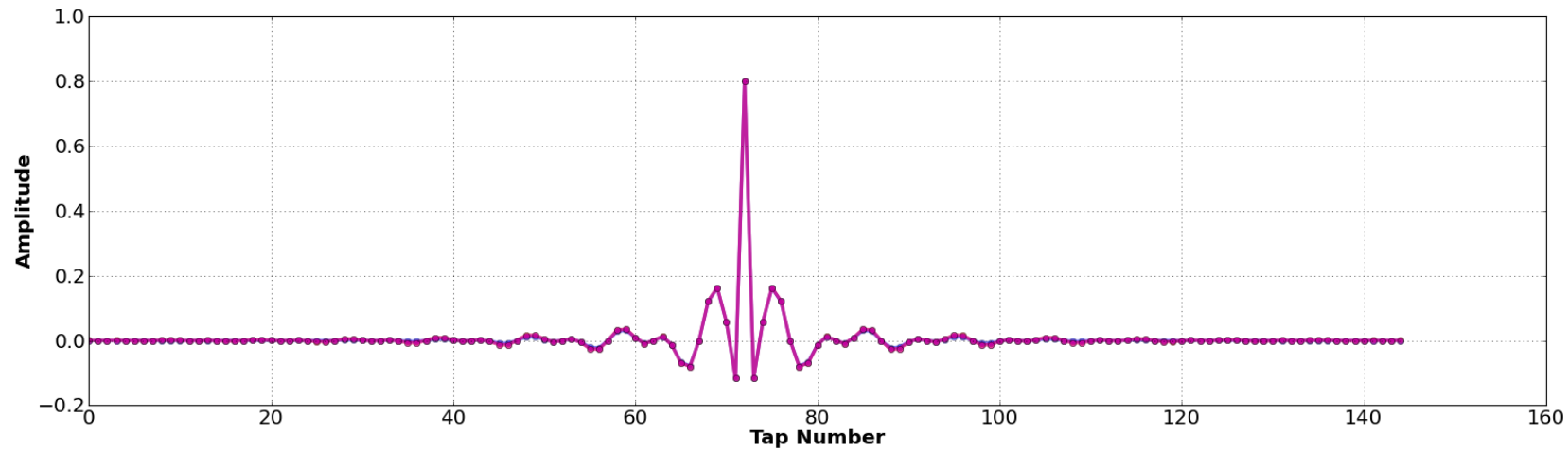
Band Pass Filter



firdes designed filters

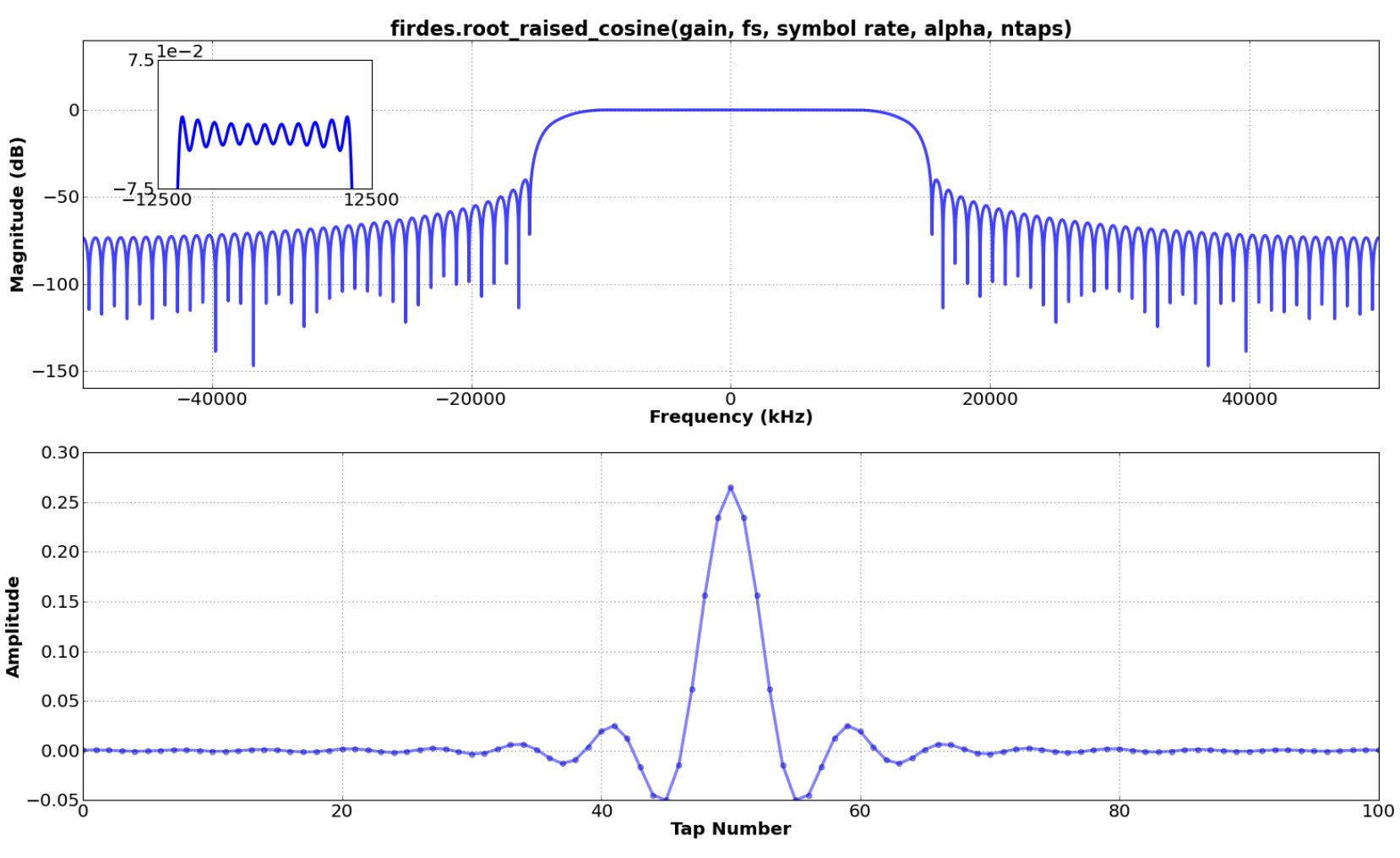
Band Reject Filter





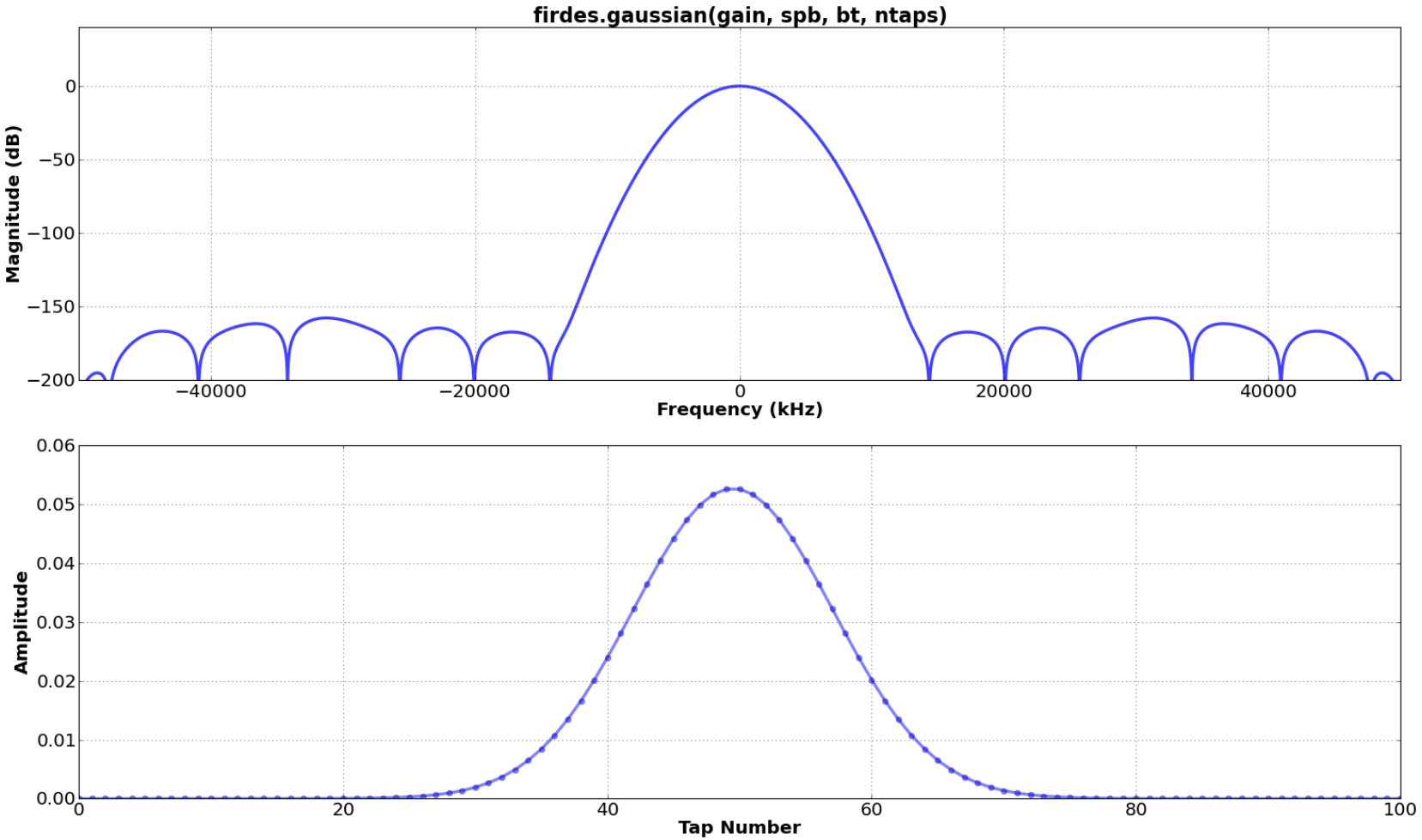
firdes designed filters

Root Raised Cosine Pulse Shapping Filter



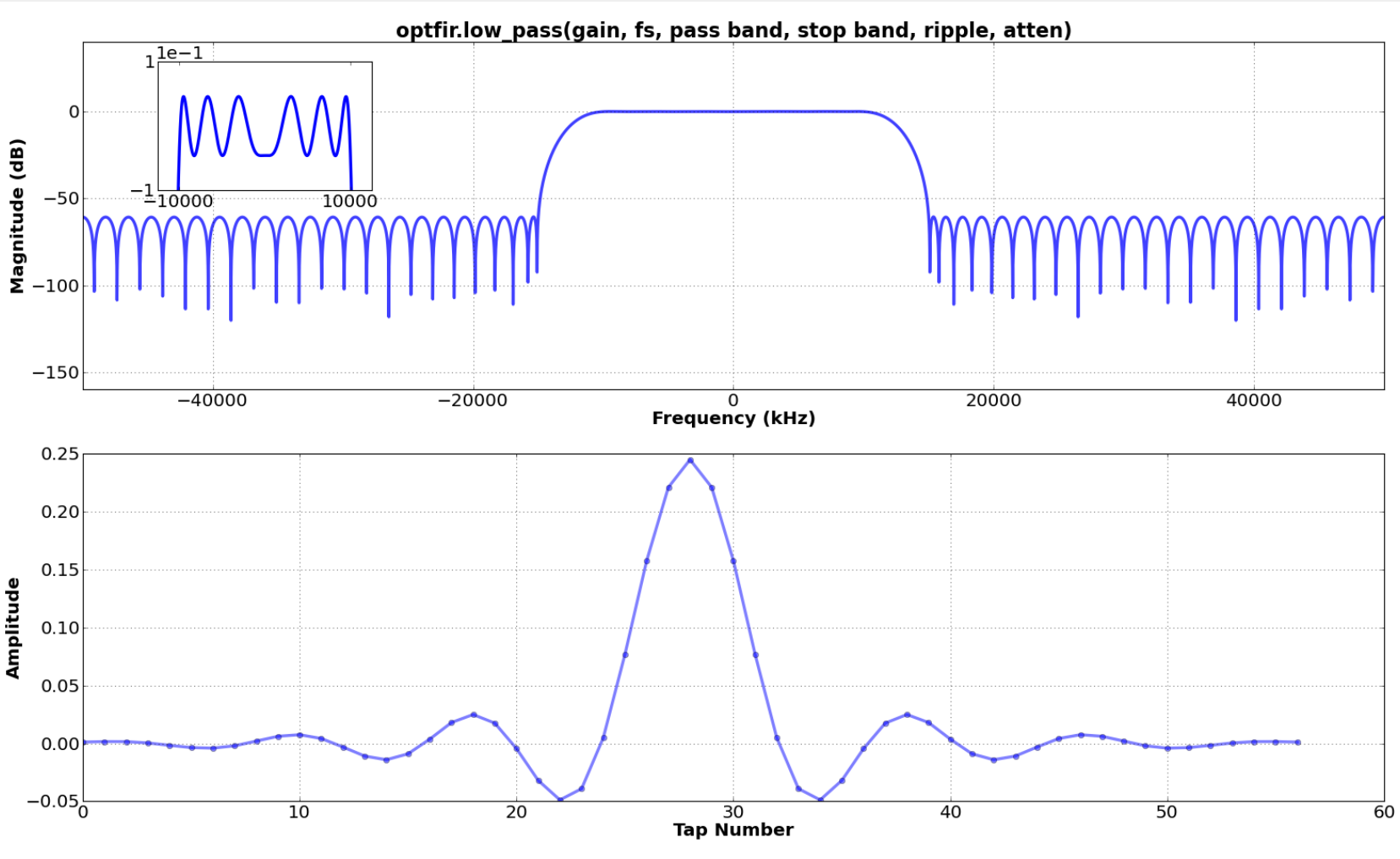
firdes designed filters

Gaussian Pulse Shapping Filter



optfir designed filter

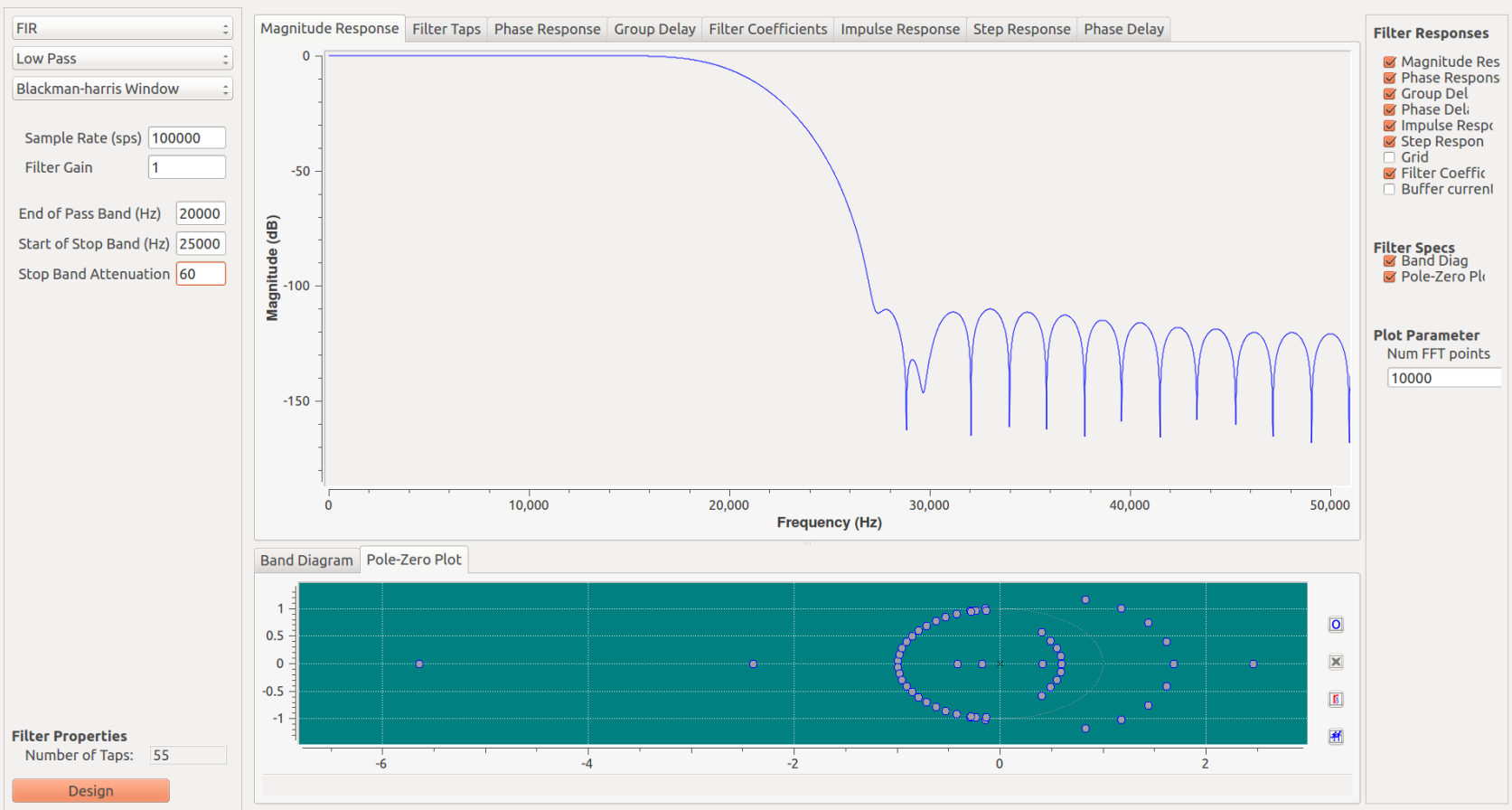
Low Pass Filter with 0.1 dB pass band ripple



gr_filter_design Tool

- Will use **firdes** or optfir depending on the filter parameters
- Can build IIR filters (requires Scipy)
- Shows designed filter in multiple types of plots
- Pole-zero plot available (and can be manipulated)
- Shows the number of taps in the filter
- Save to a CSV file for later use
- Can be called live from Python, with optional attached callback

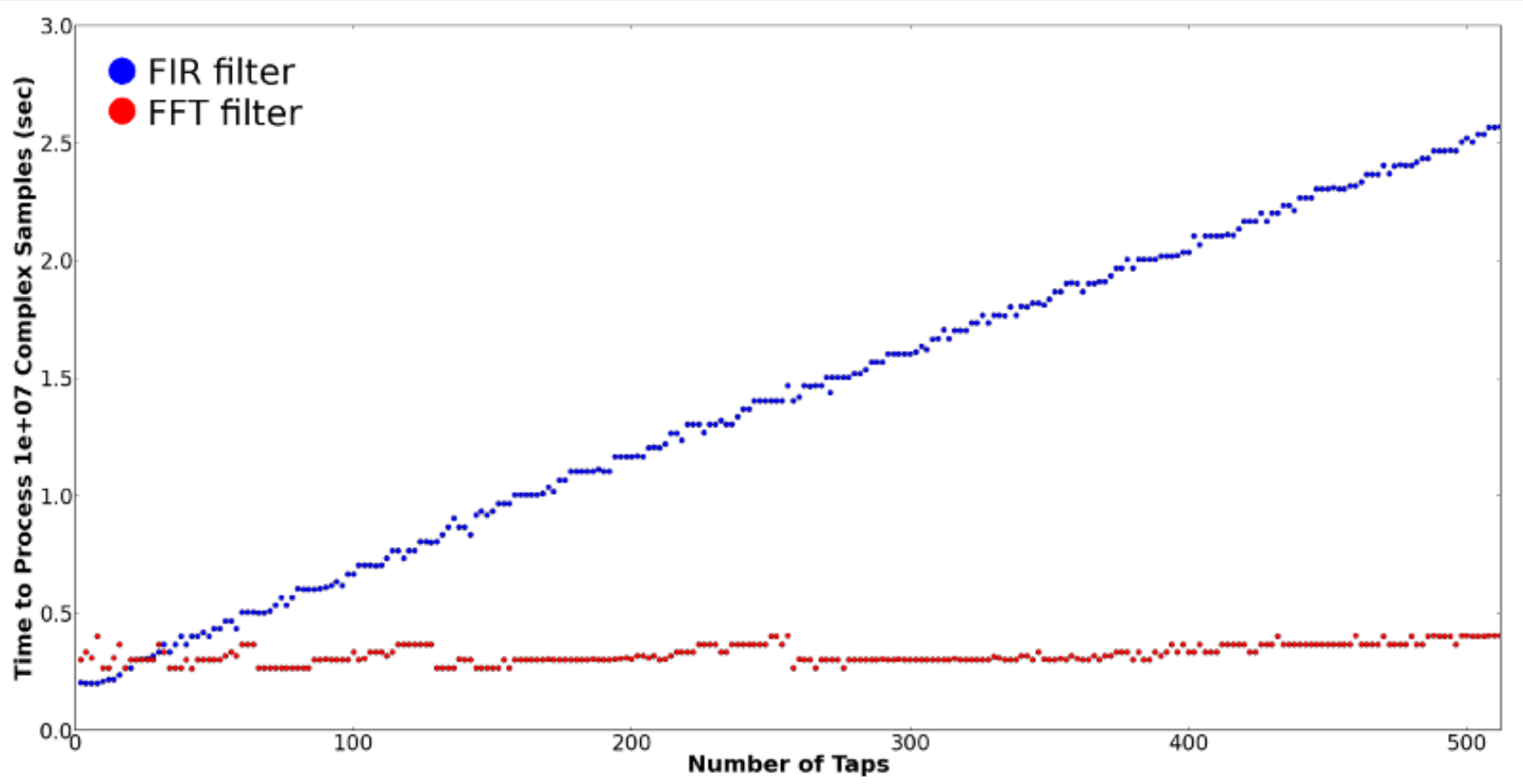
gr_filter_design



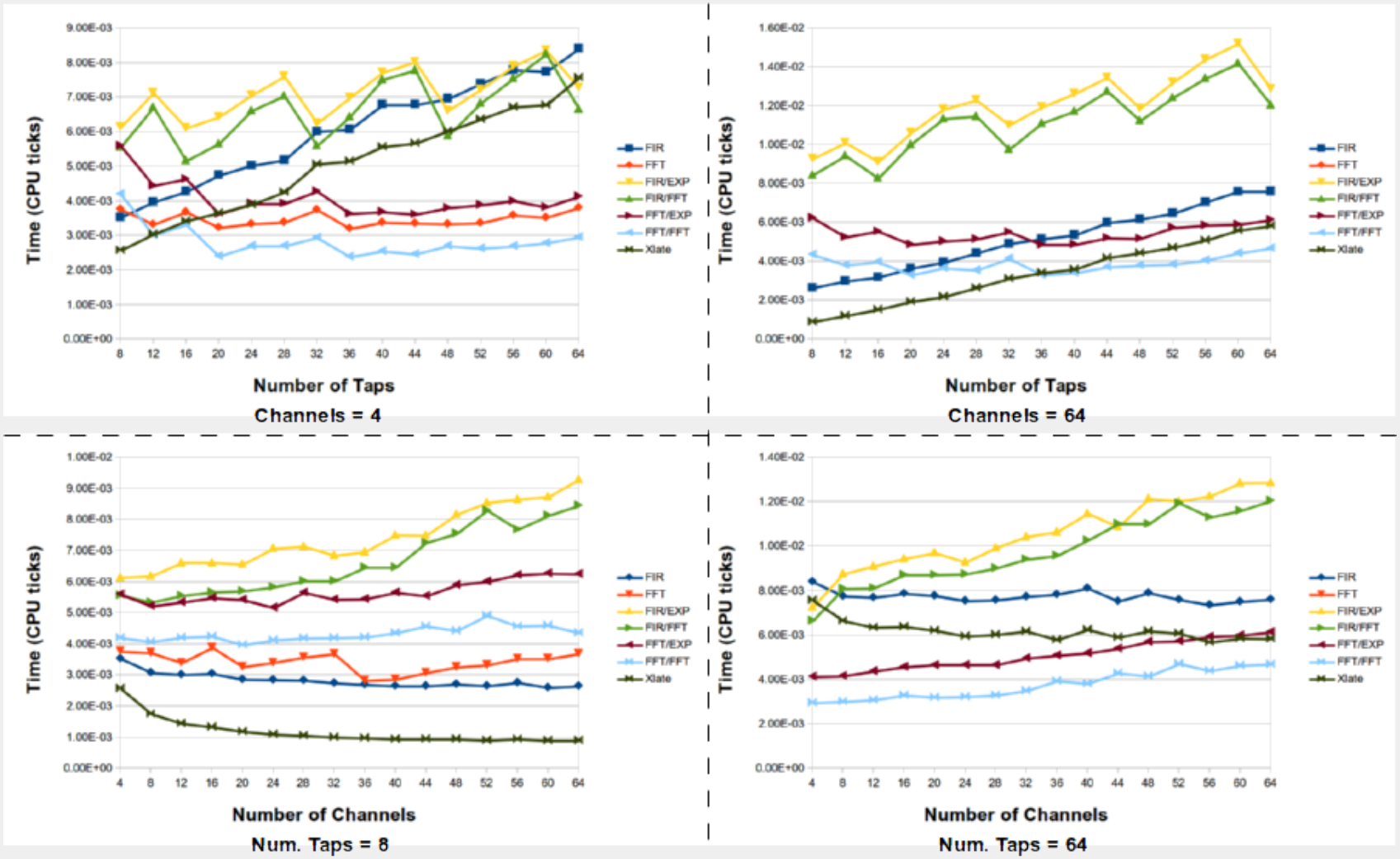
Using FIR Filters in GNU Radio

- from gnuradio import filter
- Standard FIR convolution:
  - Performs time domain convolution (sliding dot product)
  - filter.fir_filter_ccf(decim, taps)
  - Heavily optimized with SIMD for multiple platforms
- Fast Convolution filters:
  - Frequency-domain convolution (overlap-and-save)
  - filter.fft_filter_ccc(decim, taps)
  - Uses FFTW for FFTs and SIMD for multiply

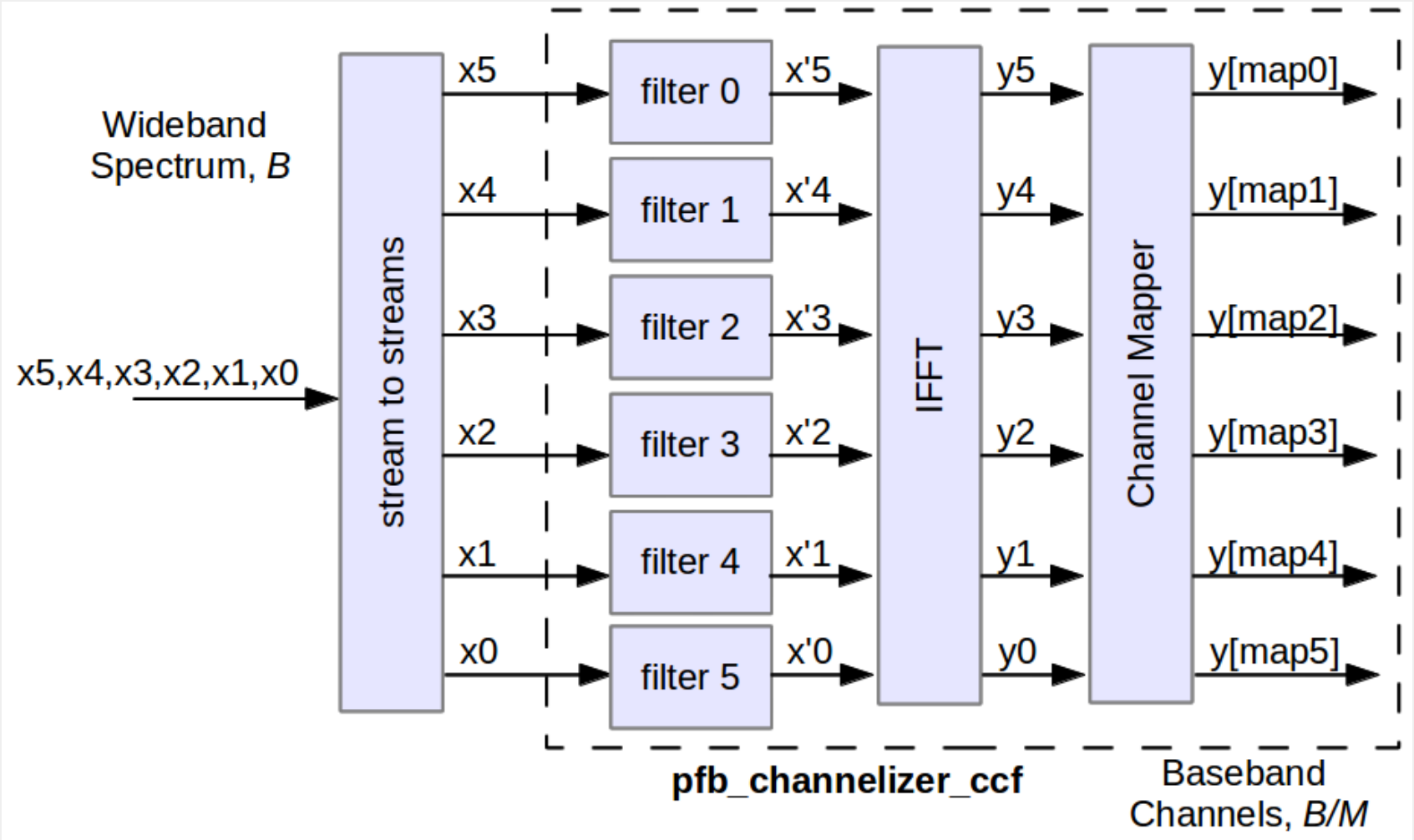
Computational Complexity of Filters



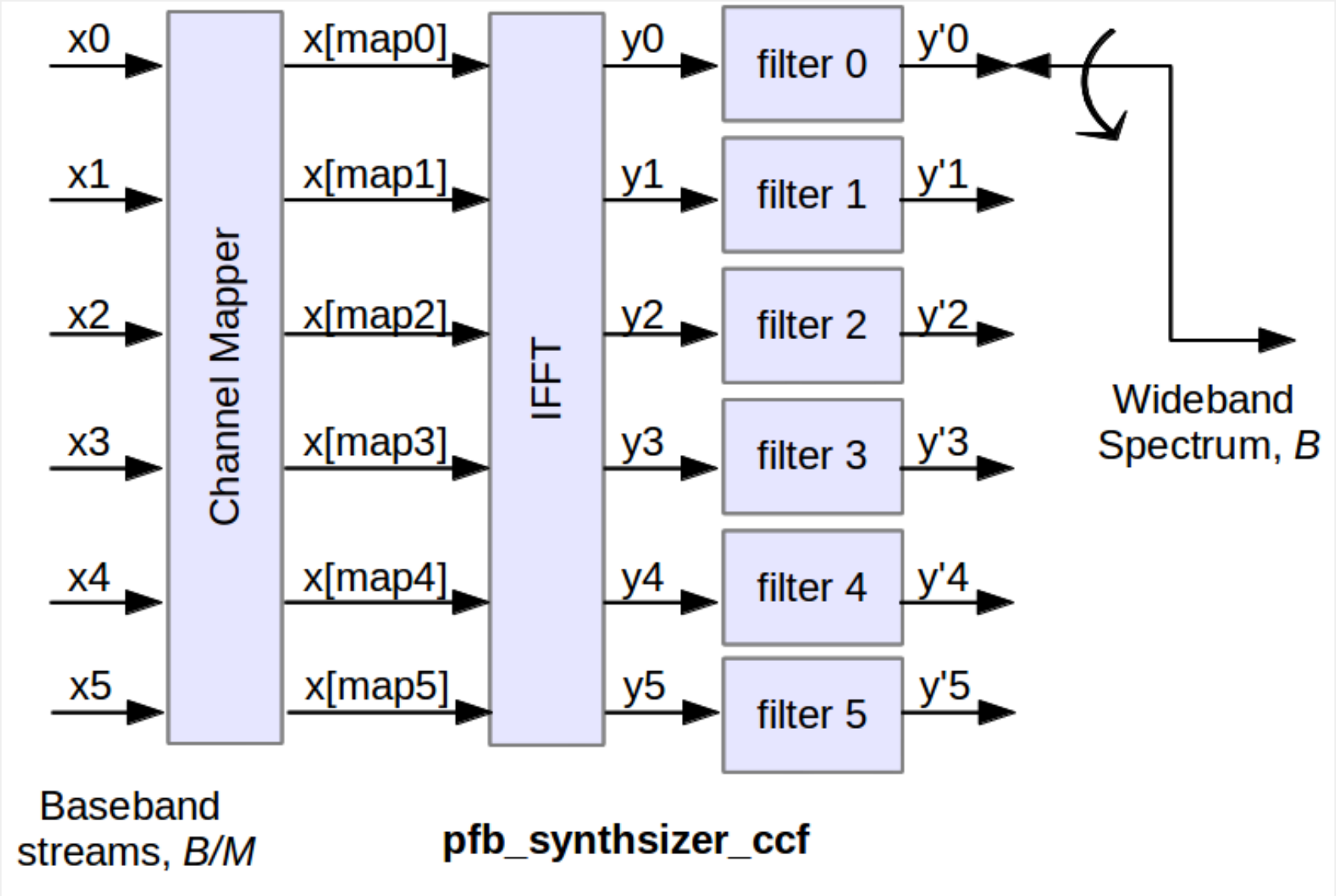
FIR vs. FFT Changes when Down-Sampling



Polyphase Channelizer



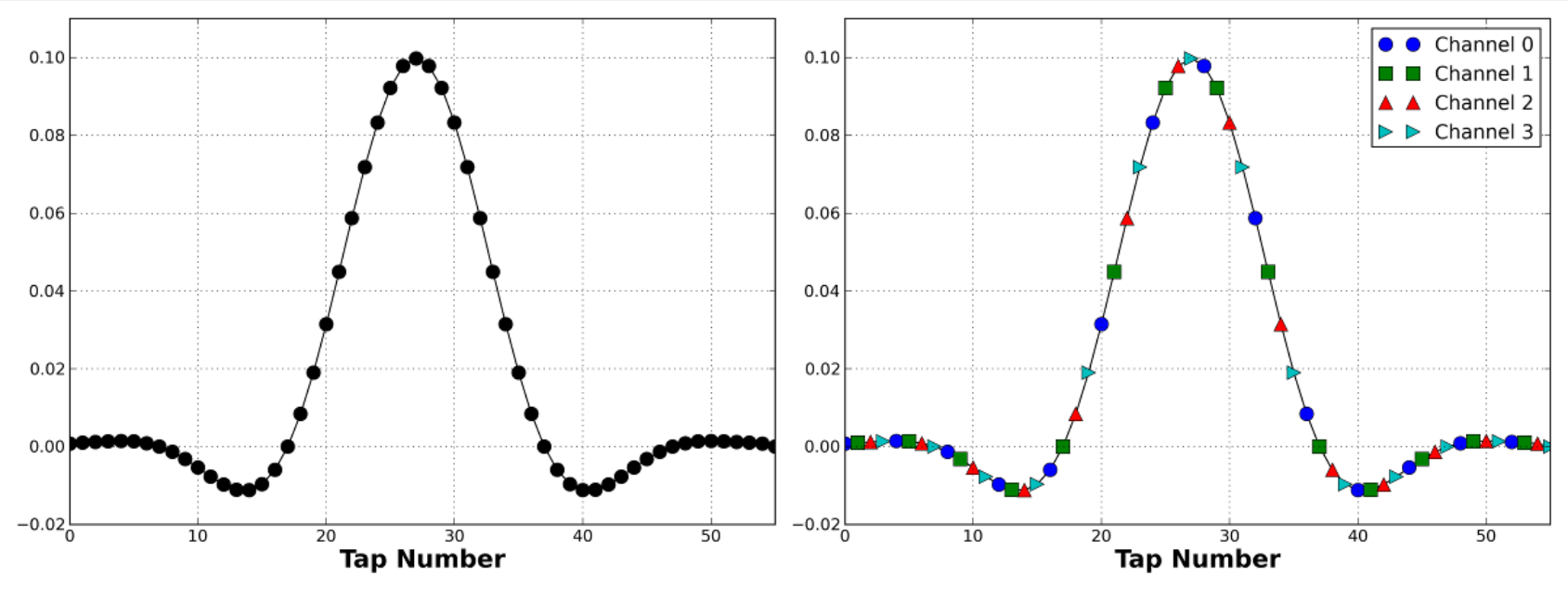
Polyphase Synthesizer



PFB Prototype Filter

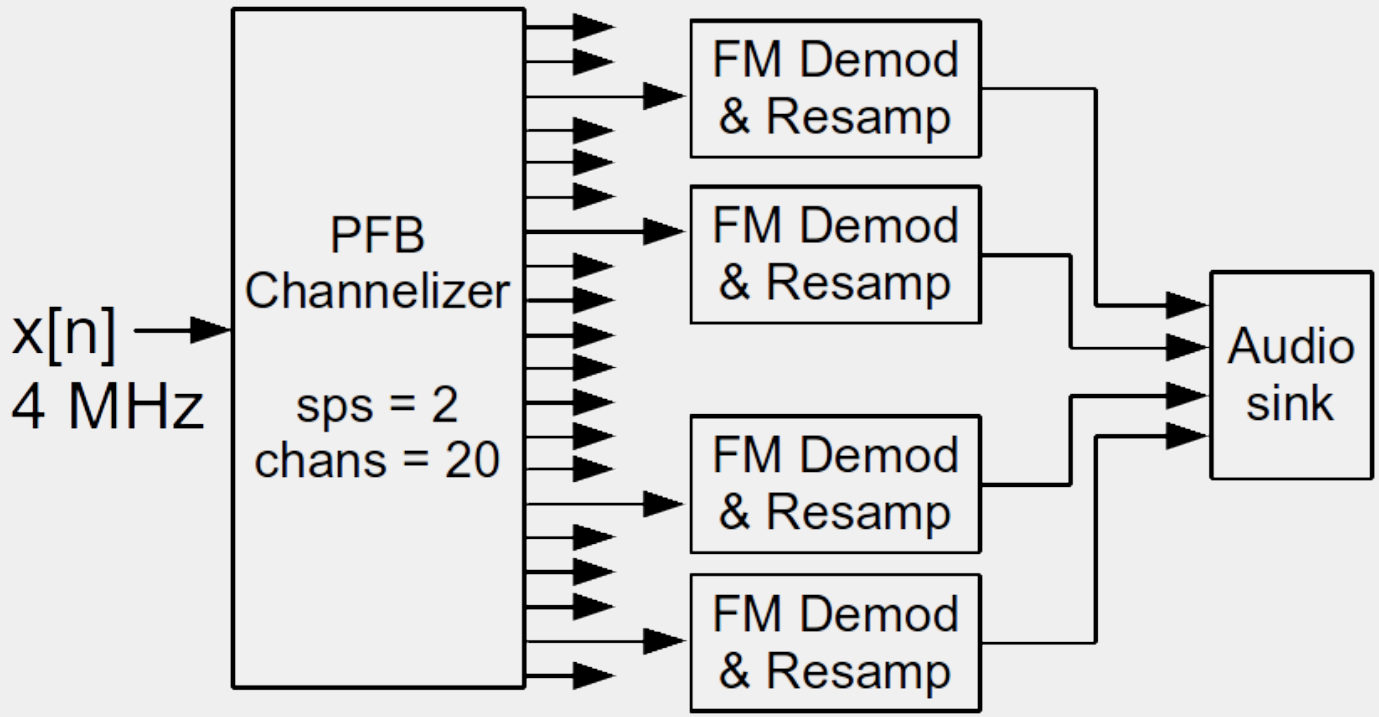
- Build the prototype filter at the highest sampling rate the filter will see.
  - Channelizer: the input sample rate
  - Synthesizer: the output sample rate
- The width of the filter is the width of the actual channels
- This creates a very large filter
- But it is partitioned among all M channels
  - each filterbank tends to be very small
  - 4 to 10 taps is usually adequate

4-Channel Example

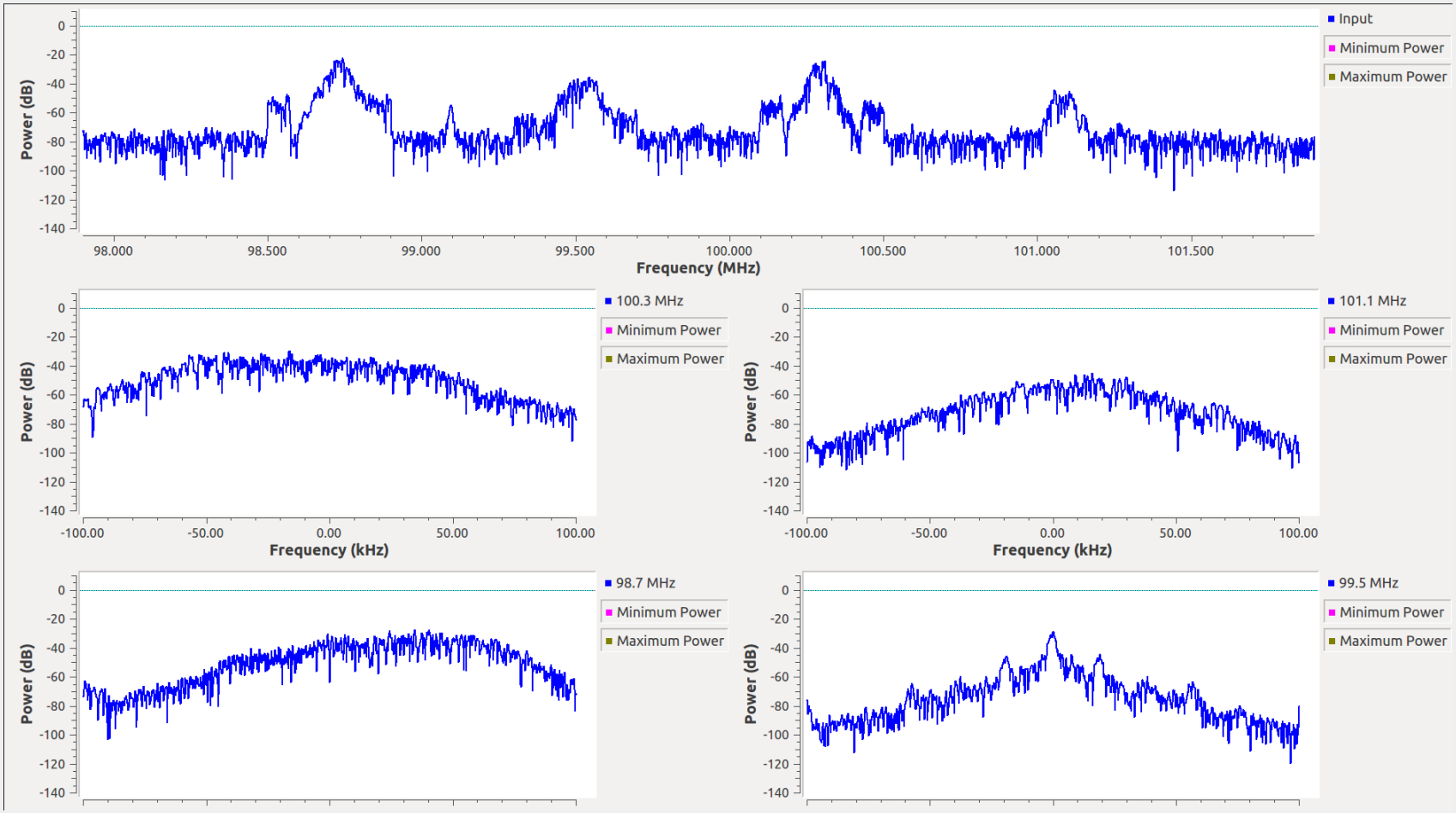


Channelizer Example: FM Band

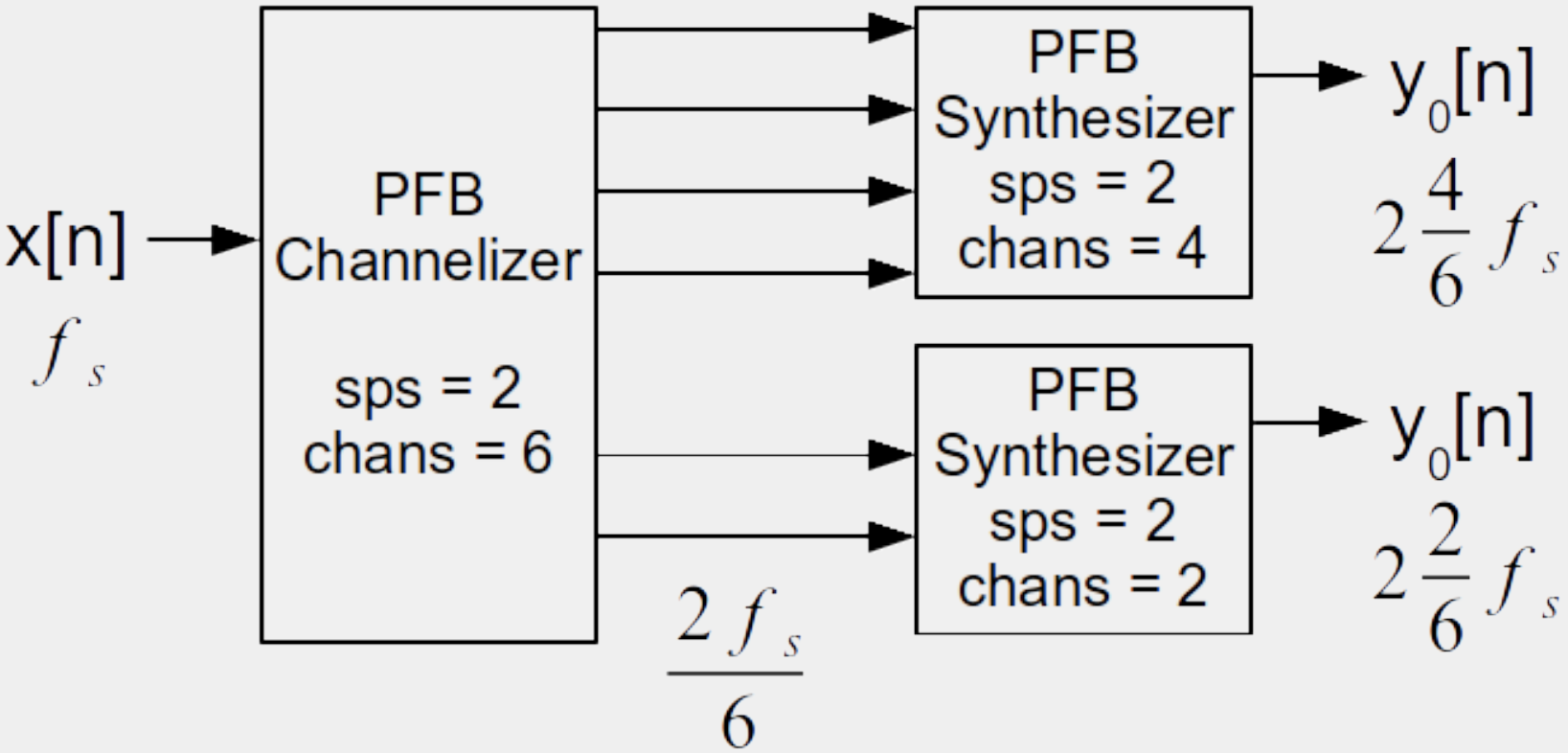
firdes.low_pass_2(1, 4M, 75k, 100k, 60, firdes.WIN_HANN)



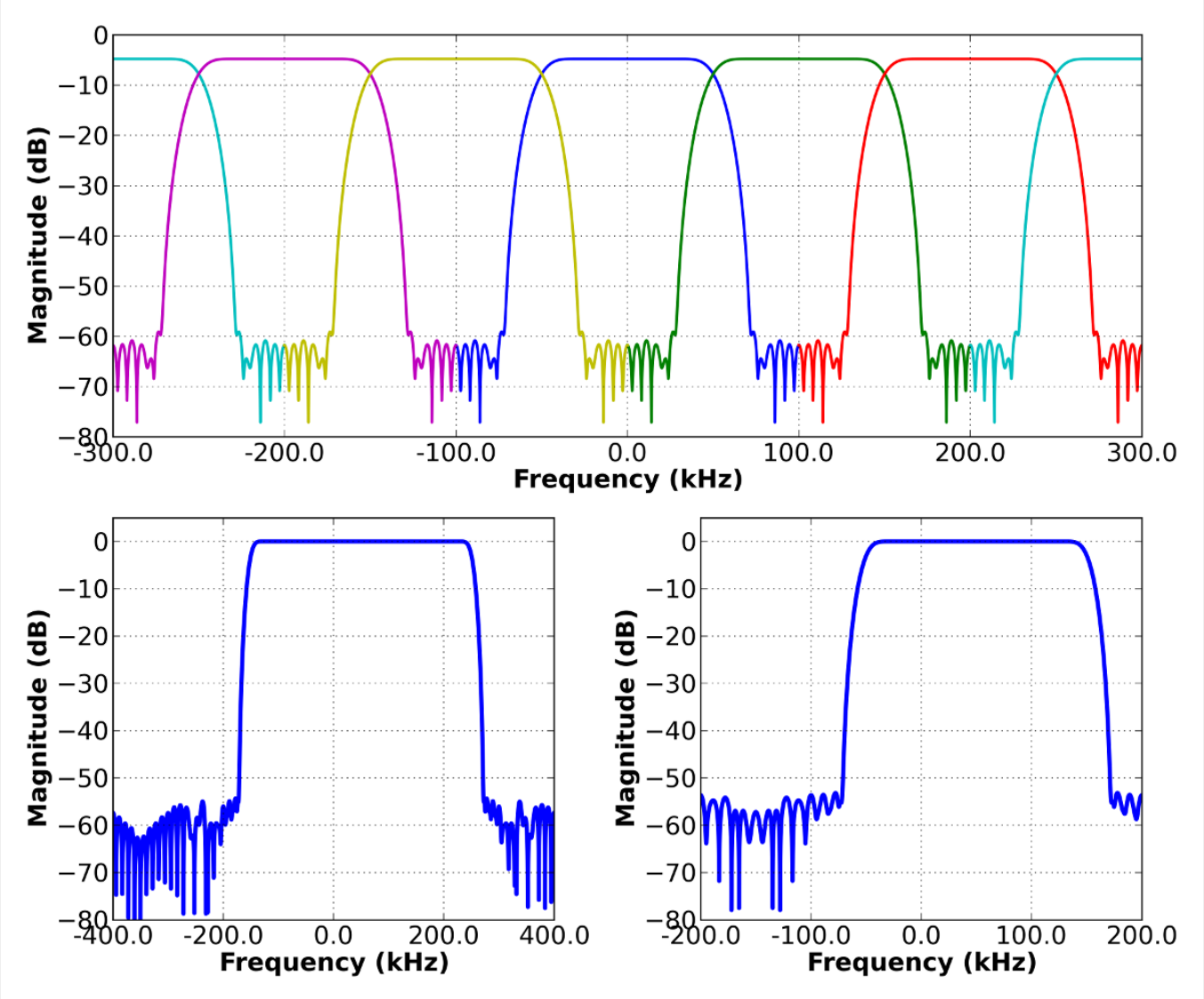
Channelizer Example: FM Band Output



Reconstruction Filterbanks



Pass an Impulse to See Distortion (what distorton?)



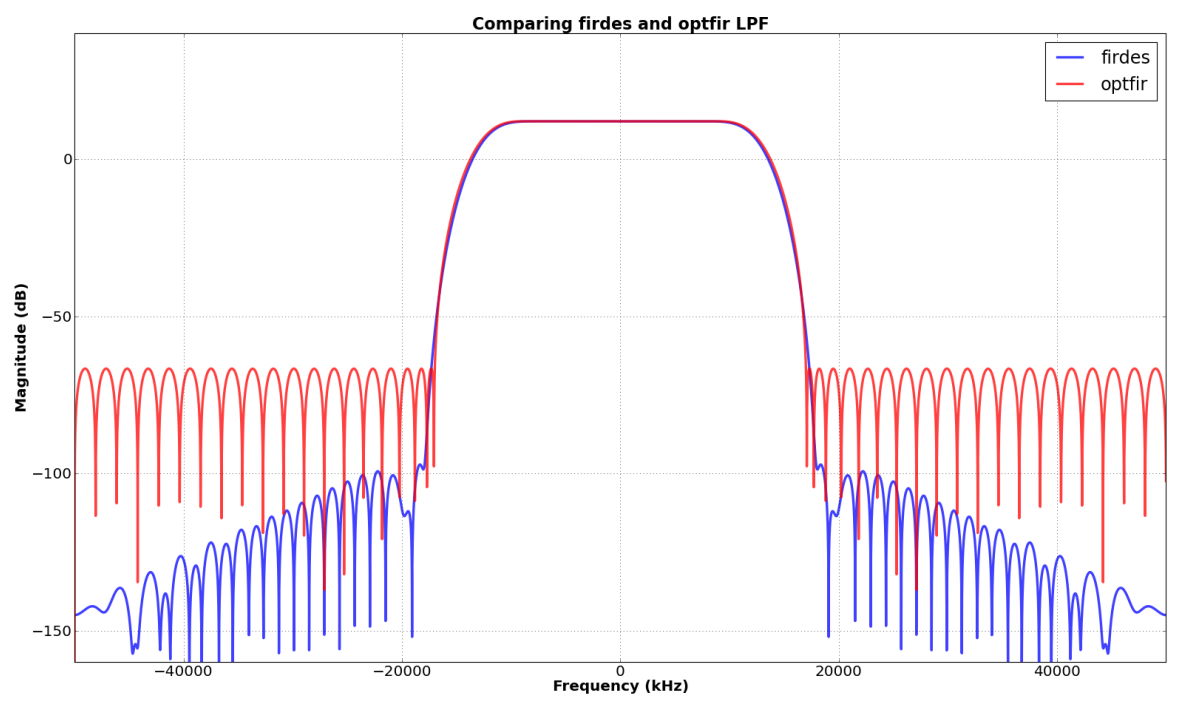
Building the Reconstruction Prototype

- Use firdes filter design tool
  - Other techniques can work; this is a guarantee
- Passband is 1/2 the channel bandwidth: -6 dB point at 0.5/fs
- Transition band is 1/5 of the channel bandwidth
  - Property discovered by Sylvain Munaut
- For M channels:
  - firdes.low_pass_2(M, M, 0.5, 0.2, 80, window)
  - window can be any window
  - I recommend Blackman-harris

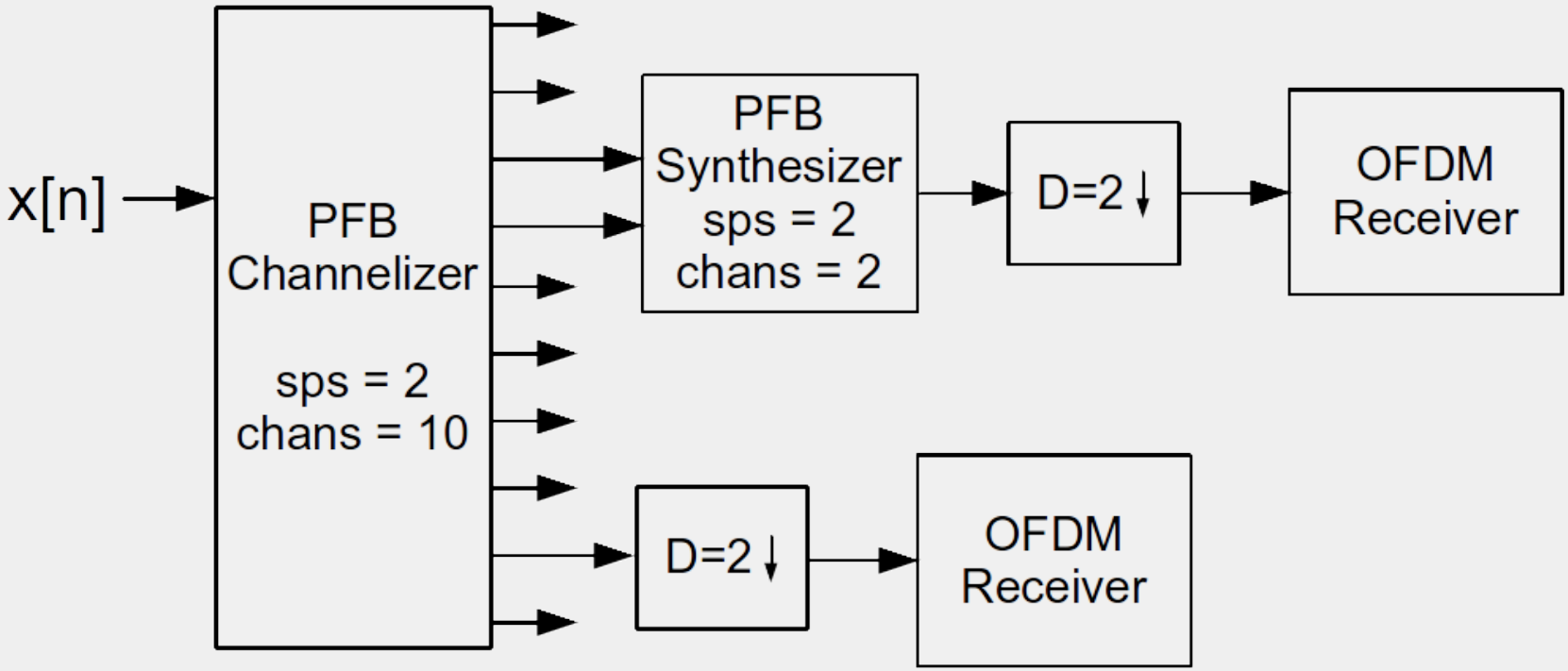
Why firdes?

M = 4

- filter.firdes.low_pass_2(M, M, 0.5, 0.2, 80, filter.firdes.WIN_BLACKMAN_hARRIS) (73 taps)
- filter.optfir.low_pass(M, M, 0.375, 0.305, 0.1, 80) (52 taps)

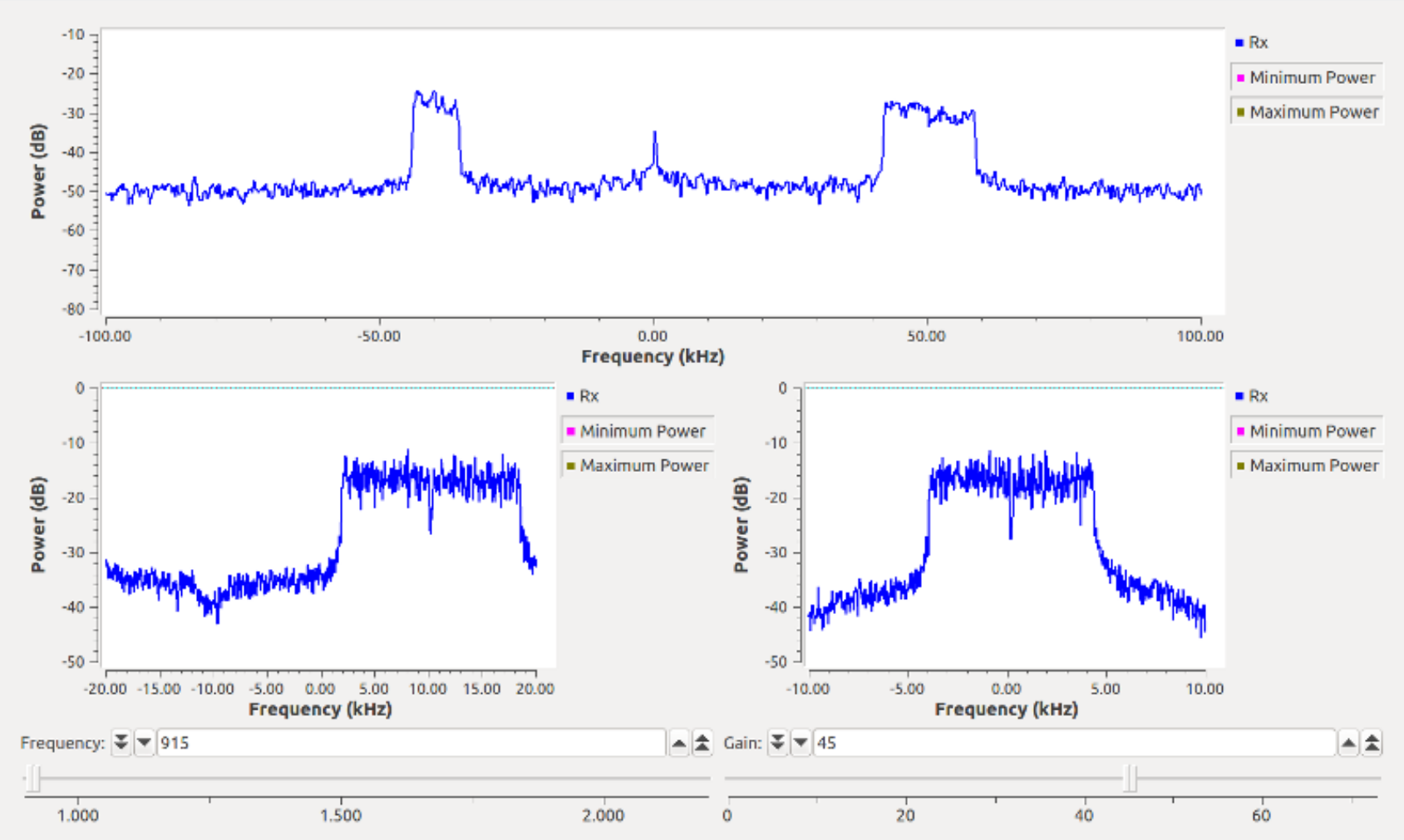


Using a Reconstruction Filterbank



Demoding Both Signals

- Received 100% of both channels



- Thank You!
- Tom Rondeau (tom@trondeau.com)
  - www.trondeau.com
  - gnuradio.org
- Matt Ettus (matt@ettus.com)
  - ettus.com