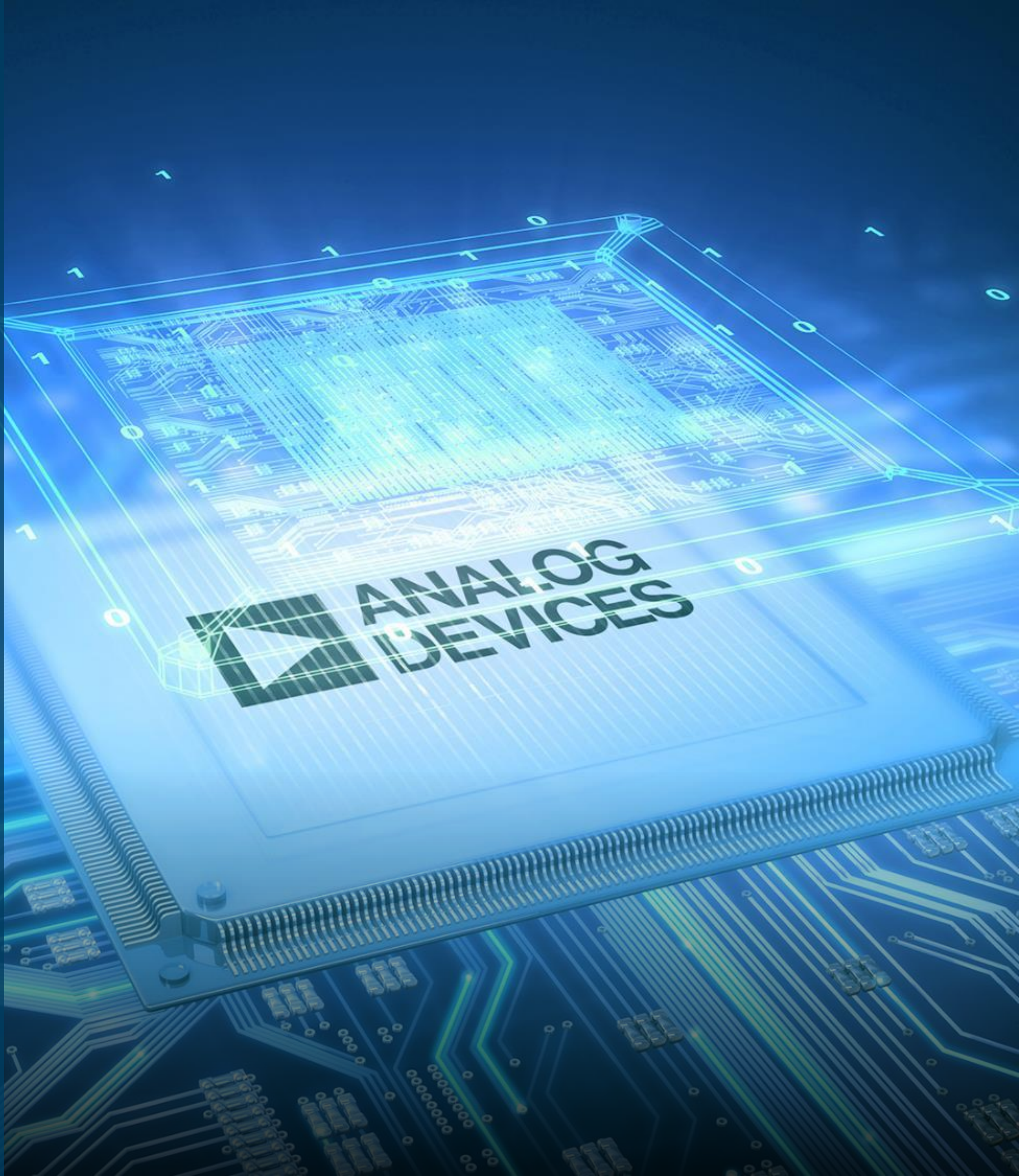
ANALOG DEVICES
AHEAD OF WHAT'S POSSIBLE™

The PHASER (CN0566)

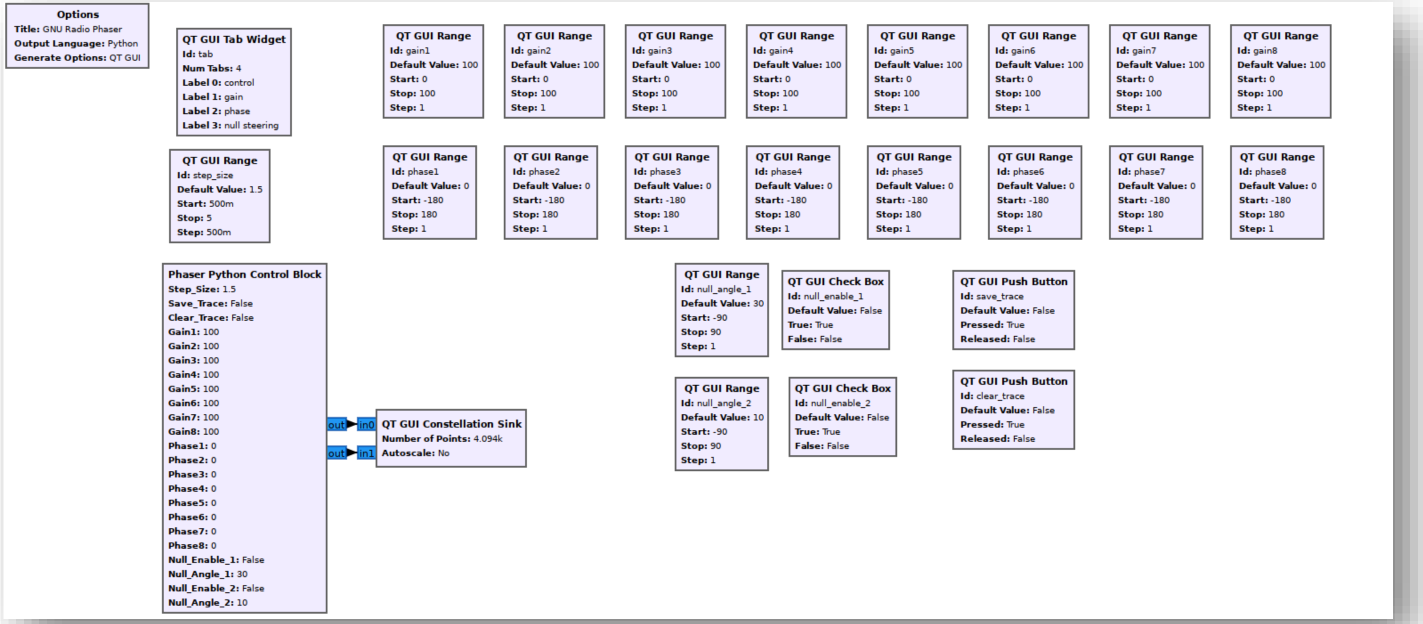
X Band Phased Array Exploration System

wiki.analog.com/phaser



Agenda

- What is “Beamforming” and where is it used?
- Introducing the Phaser X Band Phased Array Kit
- How to Control the Phaser with **Python**
- How to Control the Phaser with **GNU Radio Companion**
- Conclusion

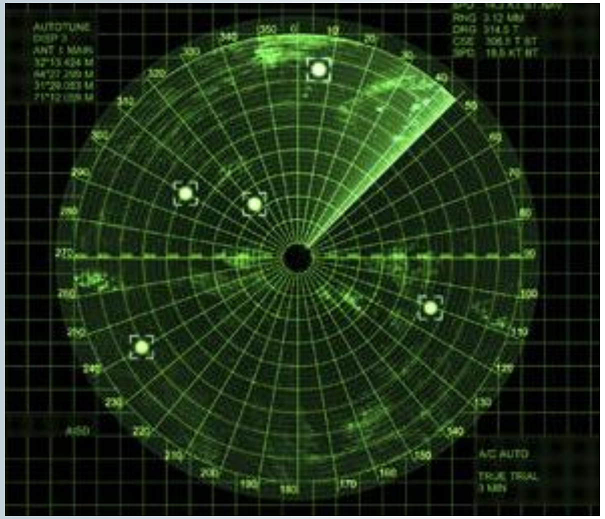


ANALOG DEVICES
AHEAD OF WHAT'S POSSIBLE™

What is beamforming?
Who uses it?
Why does it matter?

What is Phased Array Beamforming?

Rotating Antennas
(mechanical gimbles)



Phased Array antennas accomplish the same, but without mechanical movement

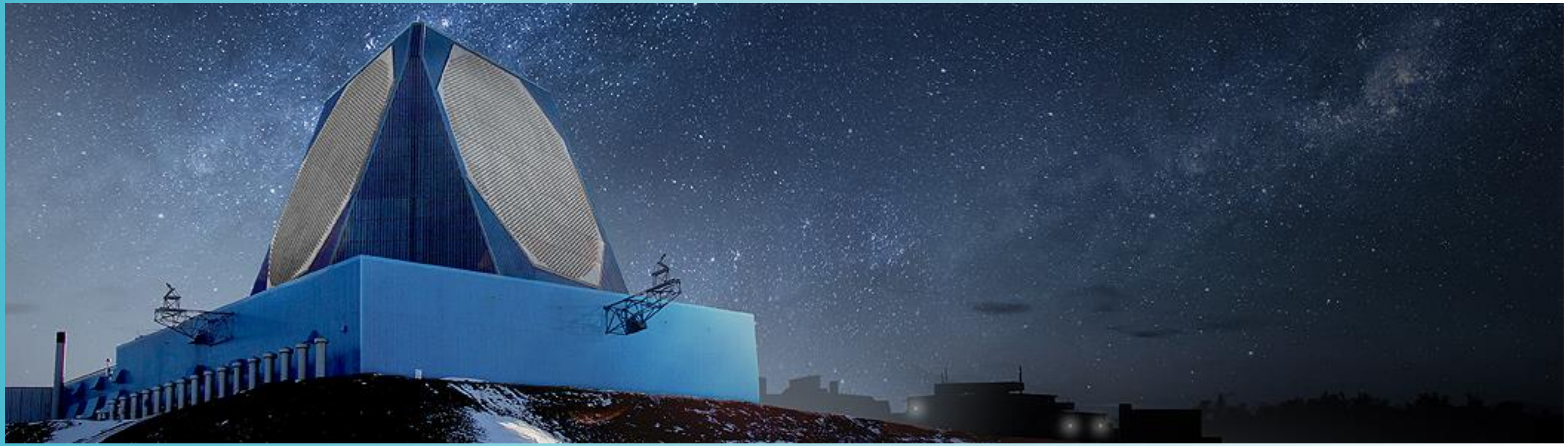
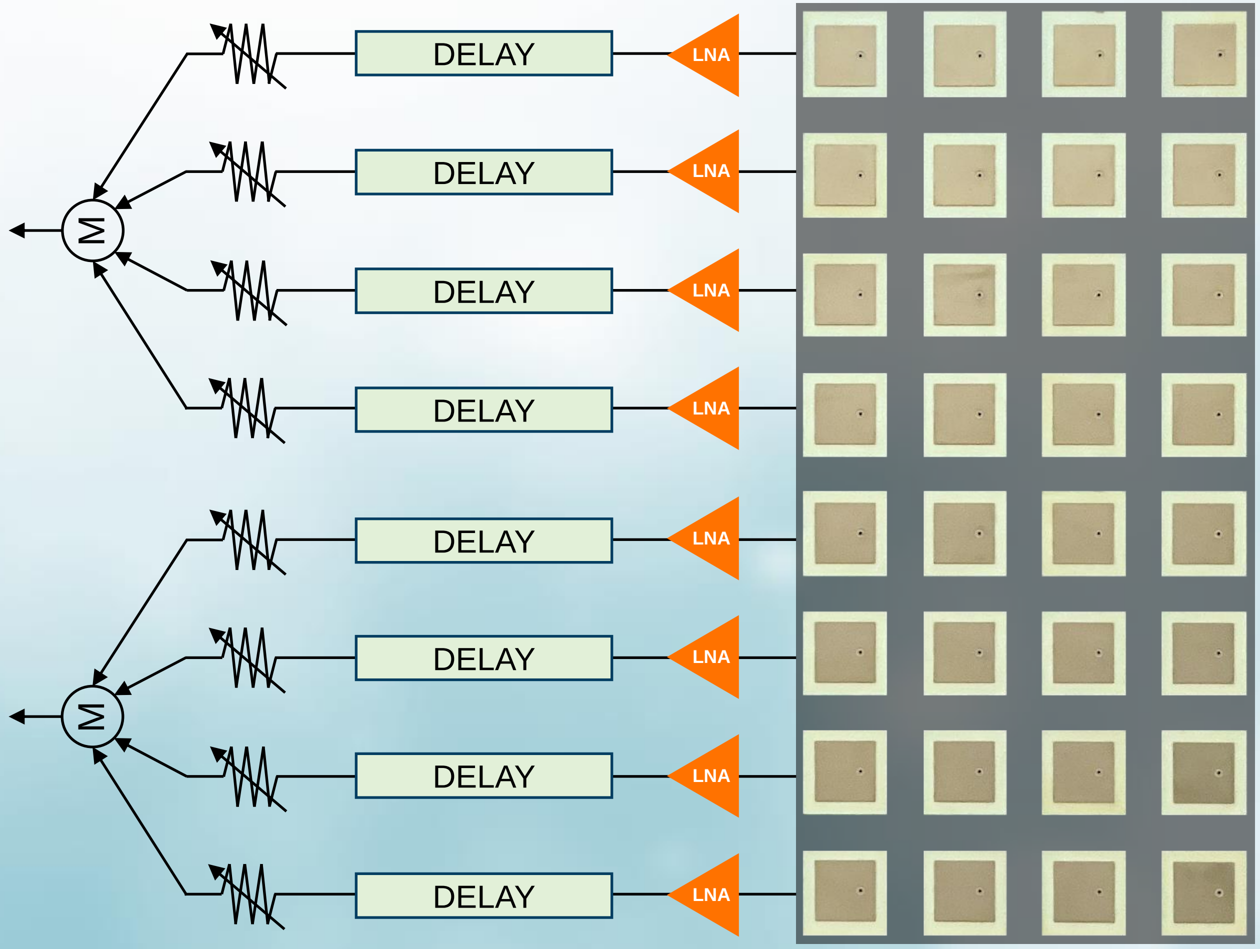
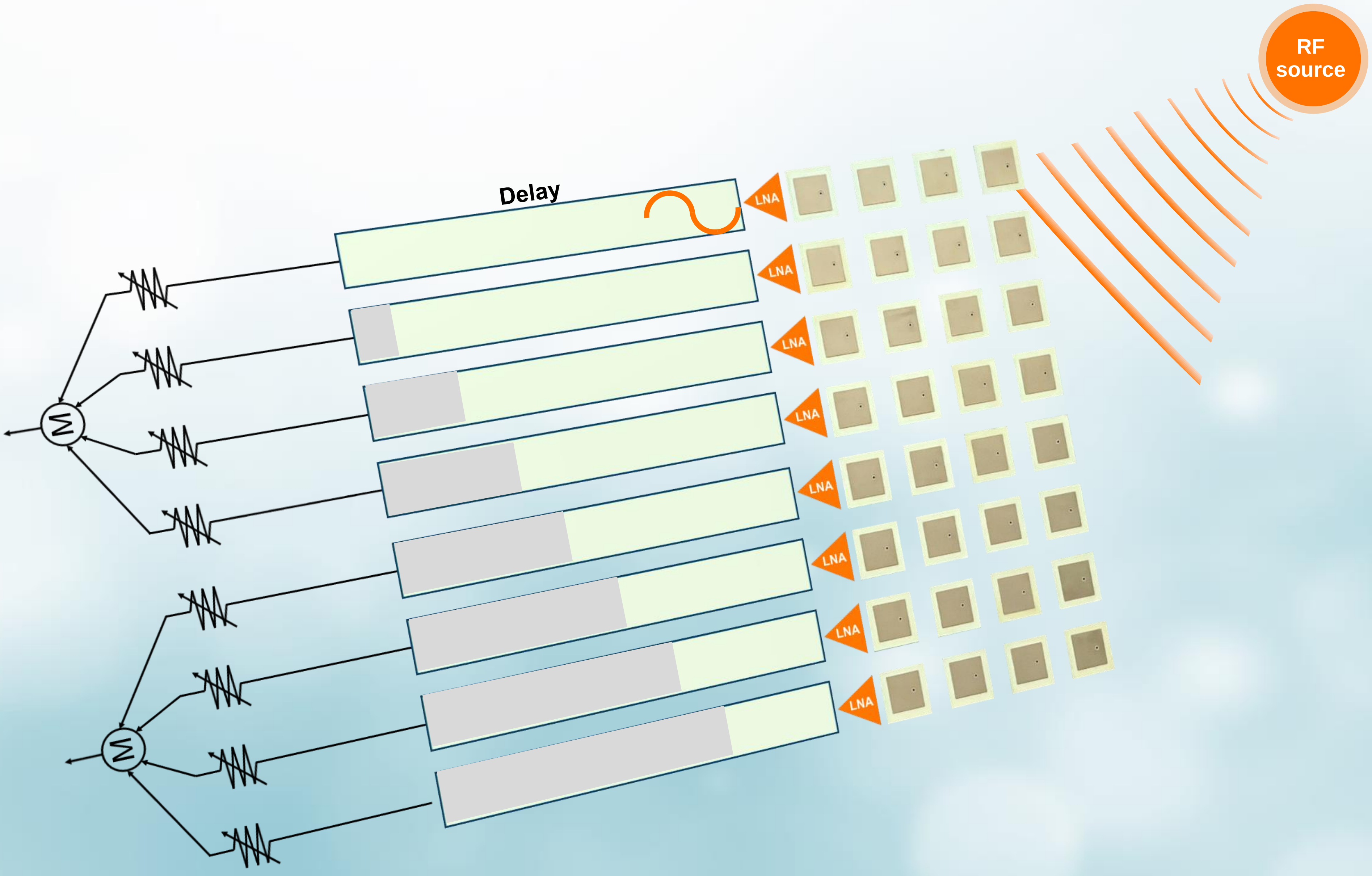


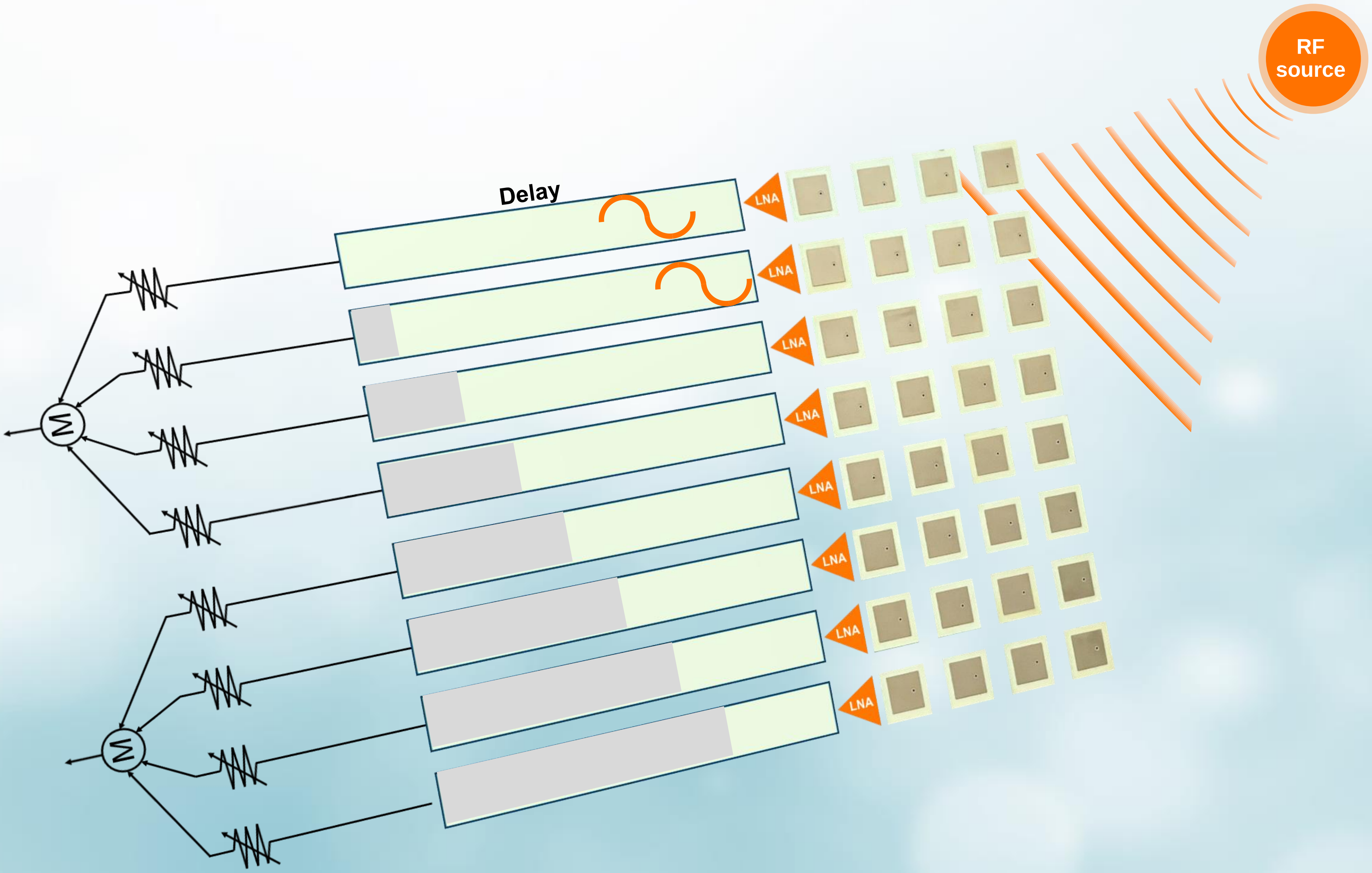
figure from https://www.analog.com/en/technical-articles/an-interview-with-analog-devices-discussing-rf-electronics-for-phased-array-applications.html

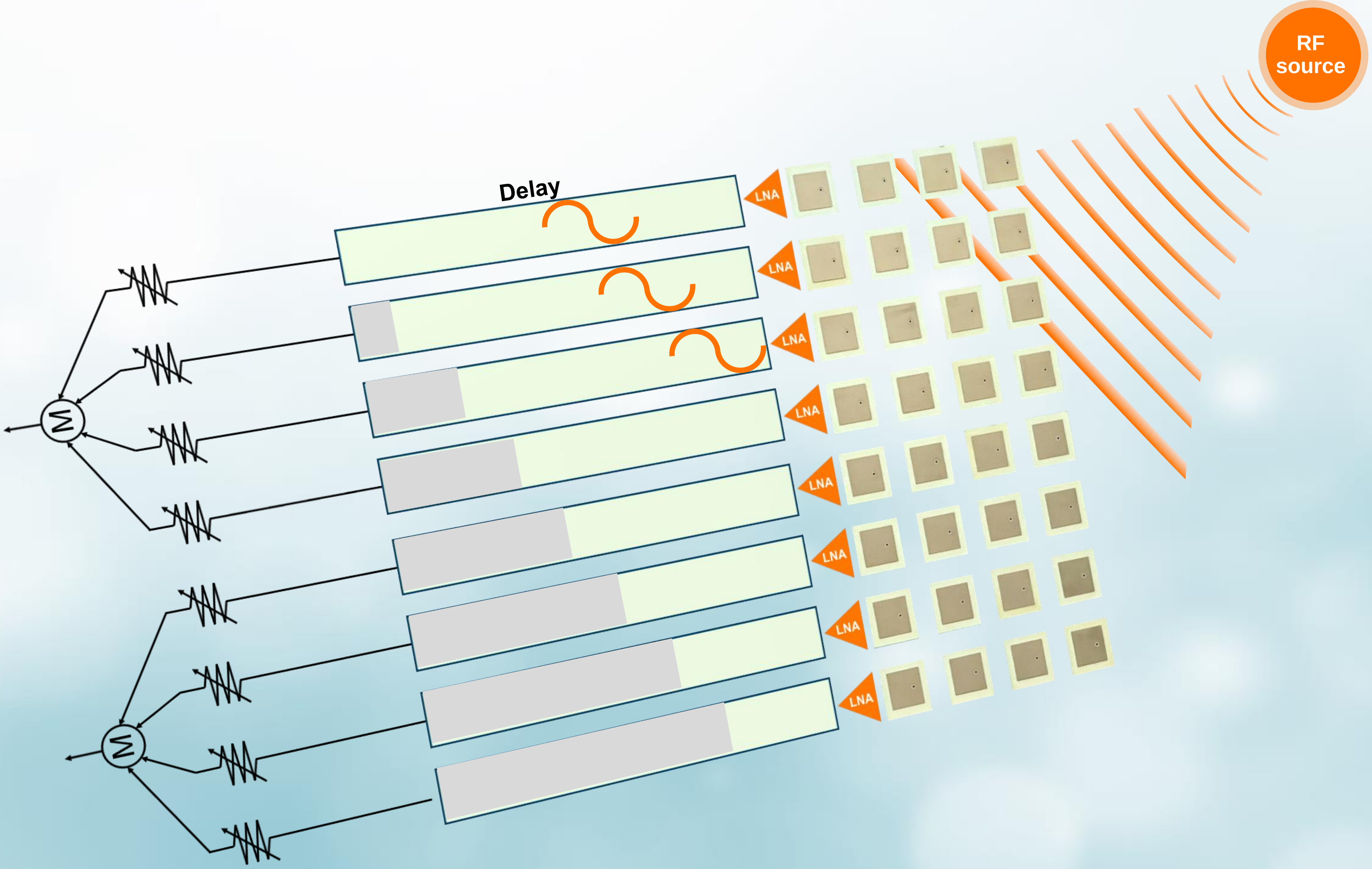
Basics of Phased Arrays



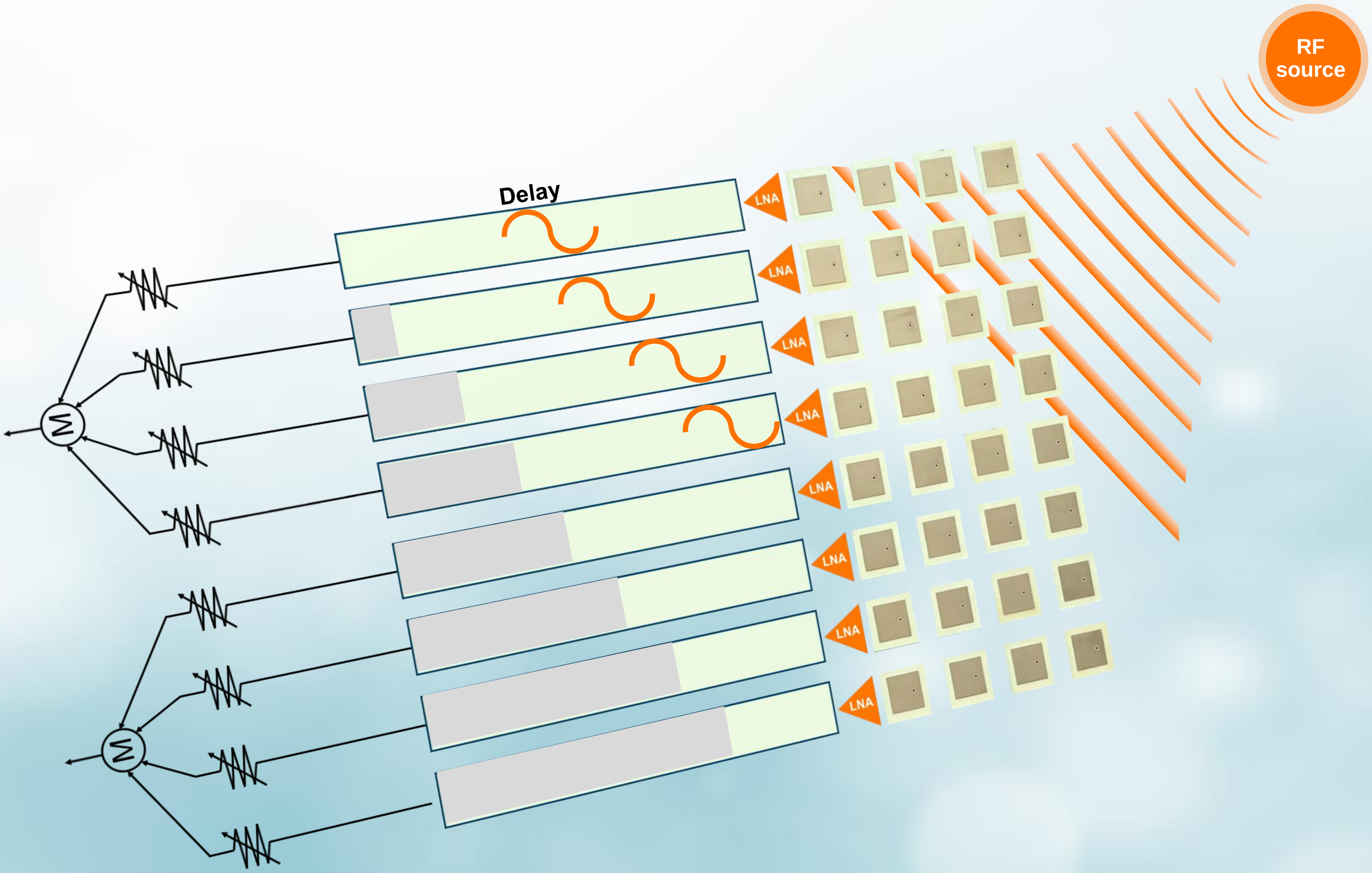


Delay
LNA
RF source

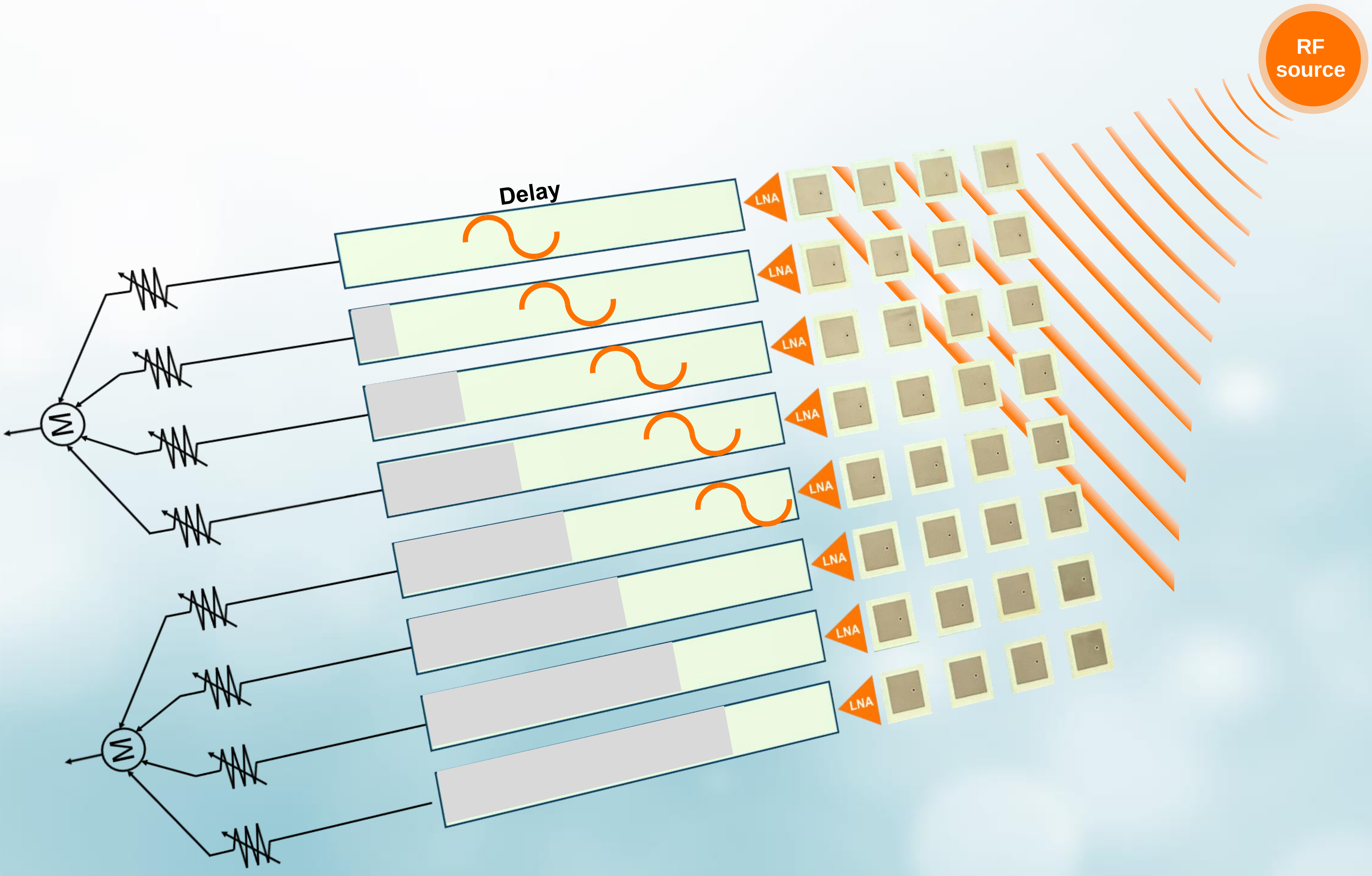

Delay
LNA
LNA
LNA
LNA
LNA
LNA
LNA
LNA
RF source

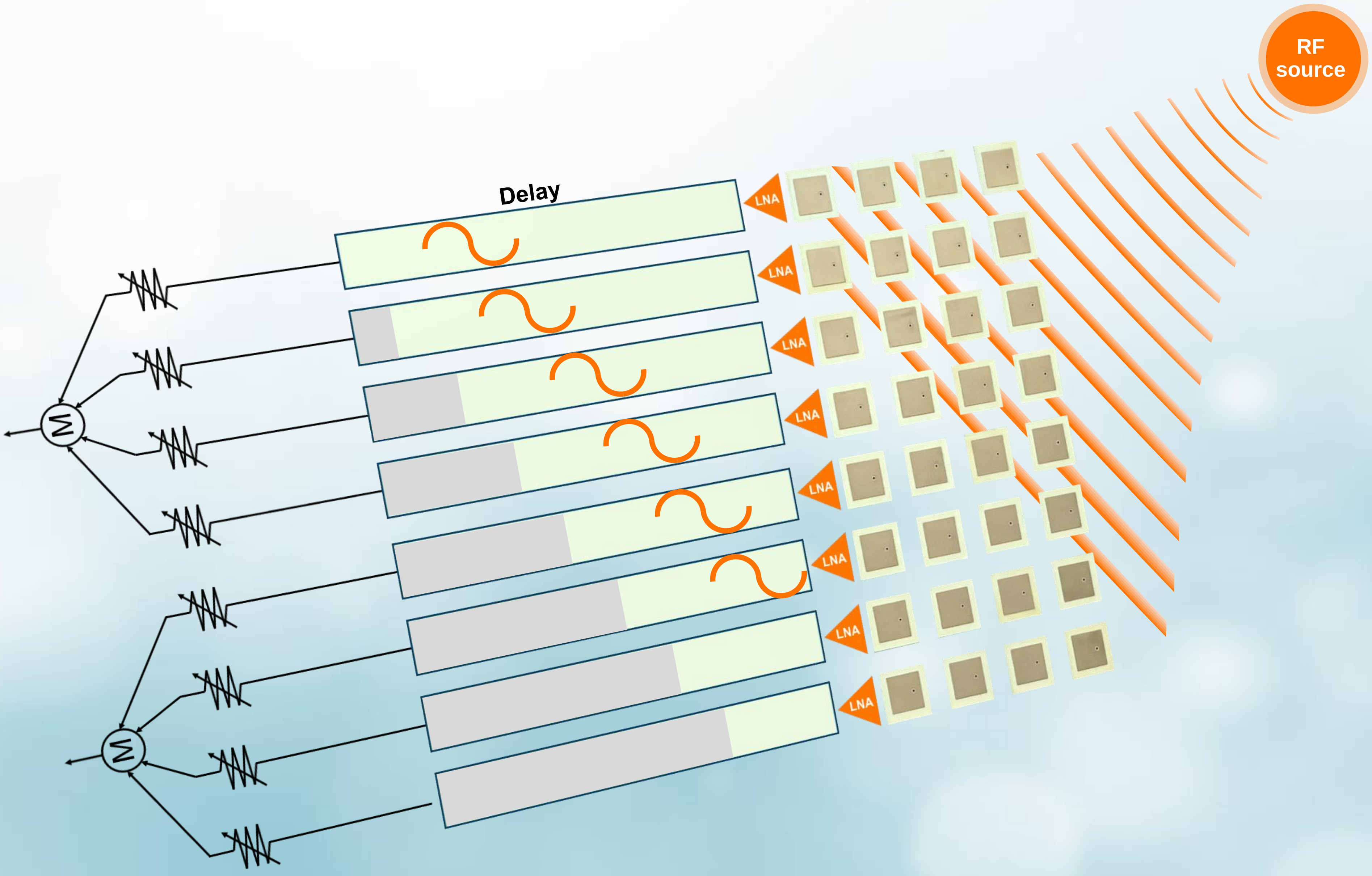


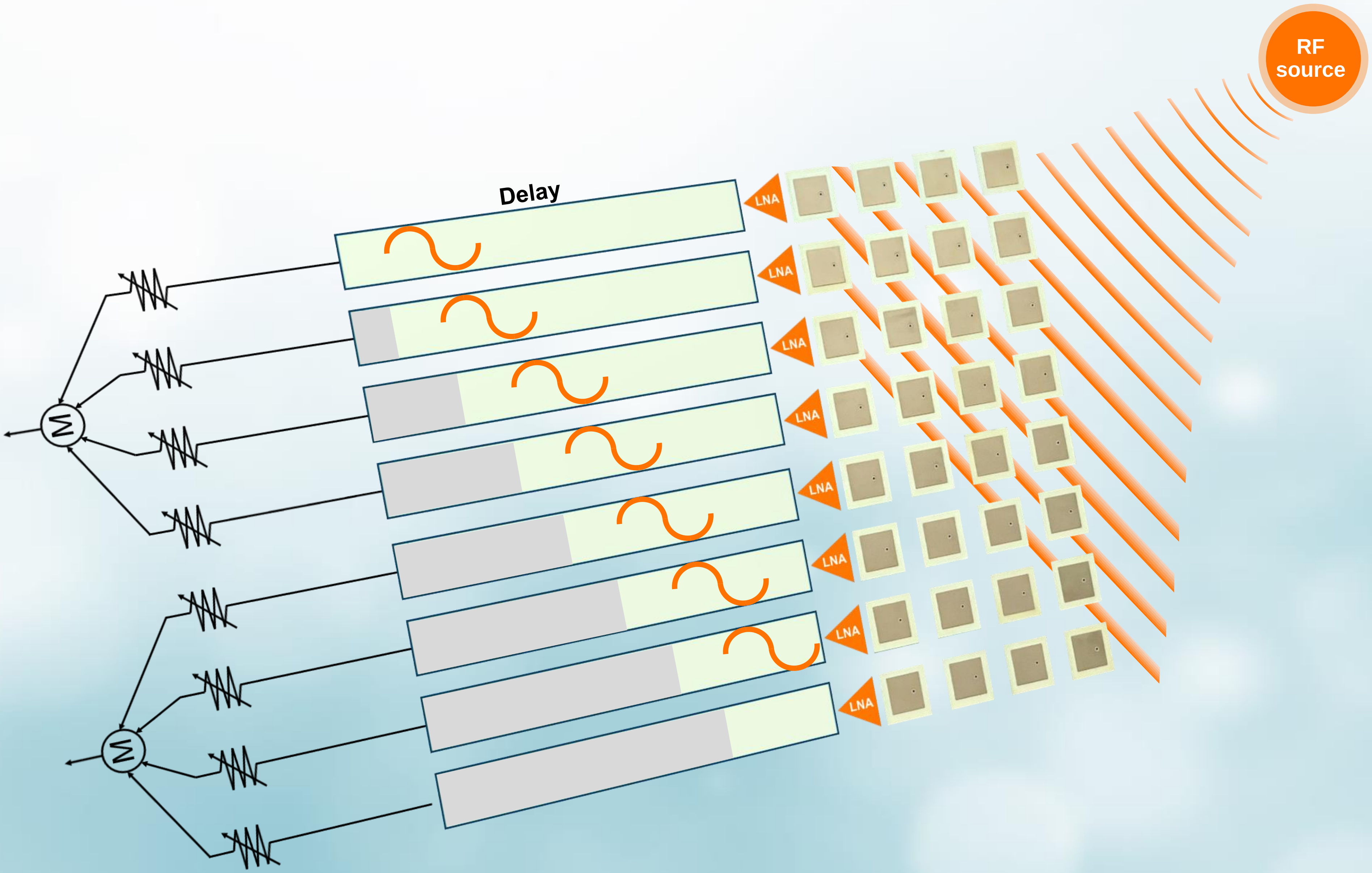
ANALOG DEVICES
AHEAD OF WHAT'S POSSIBLE™

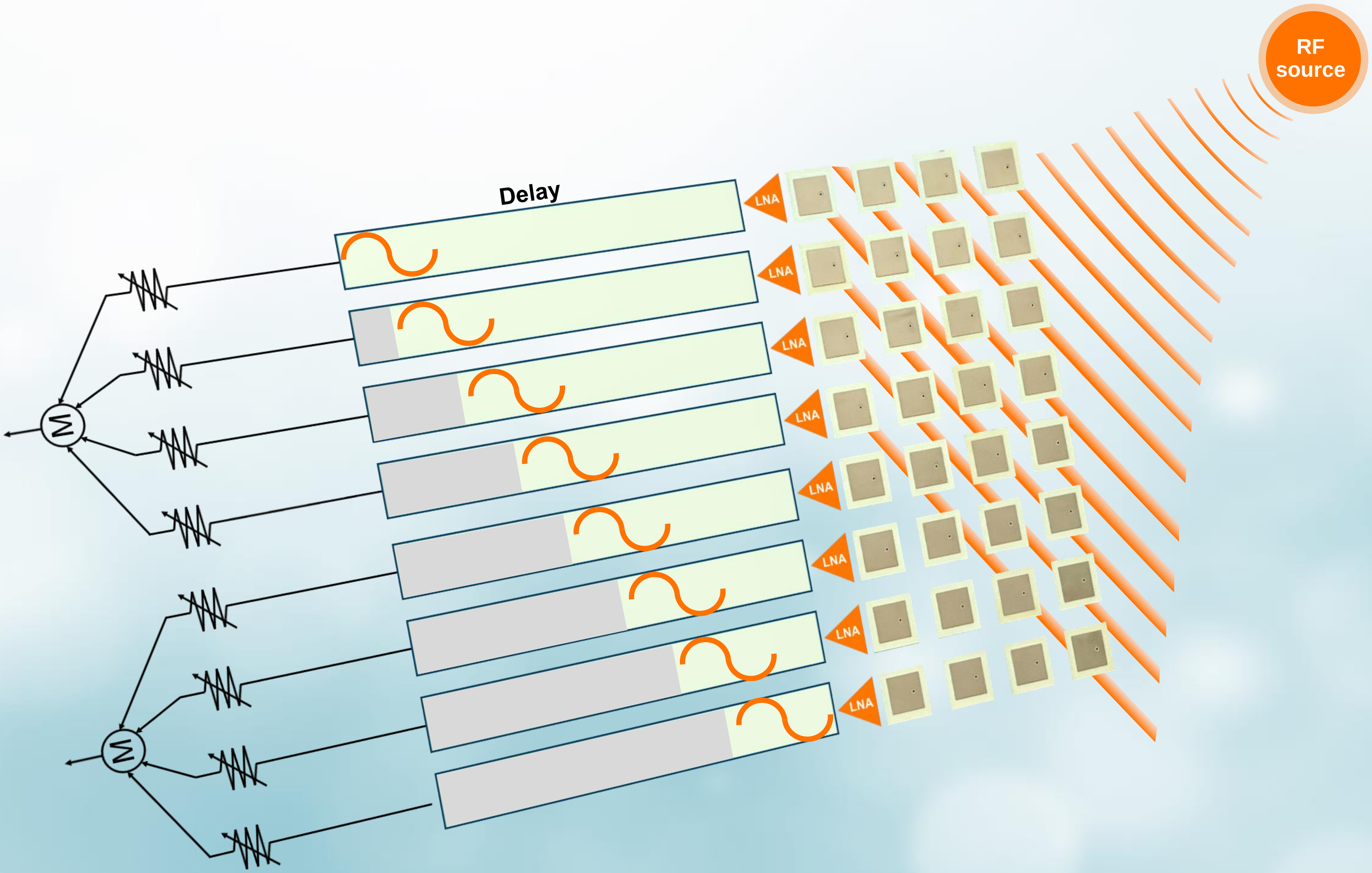


ANALOG DEVICES
AHEAD OF WHAT'S POSSIBLE™

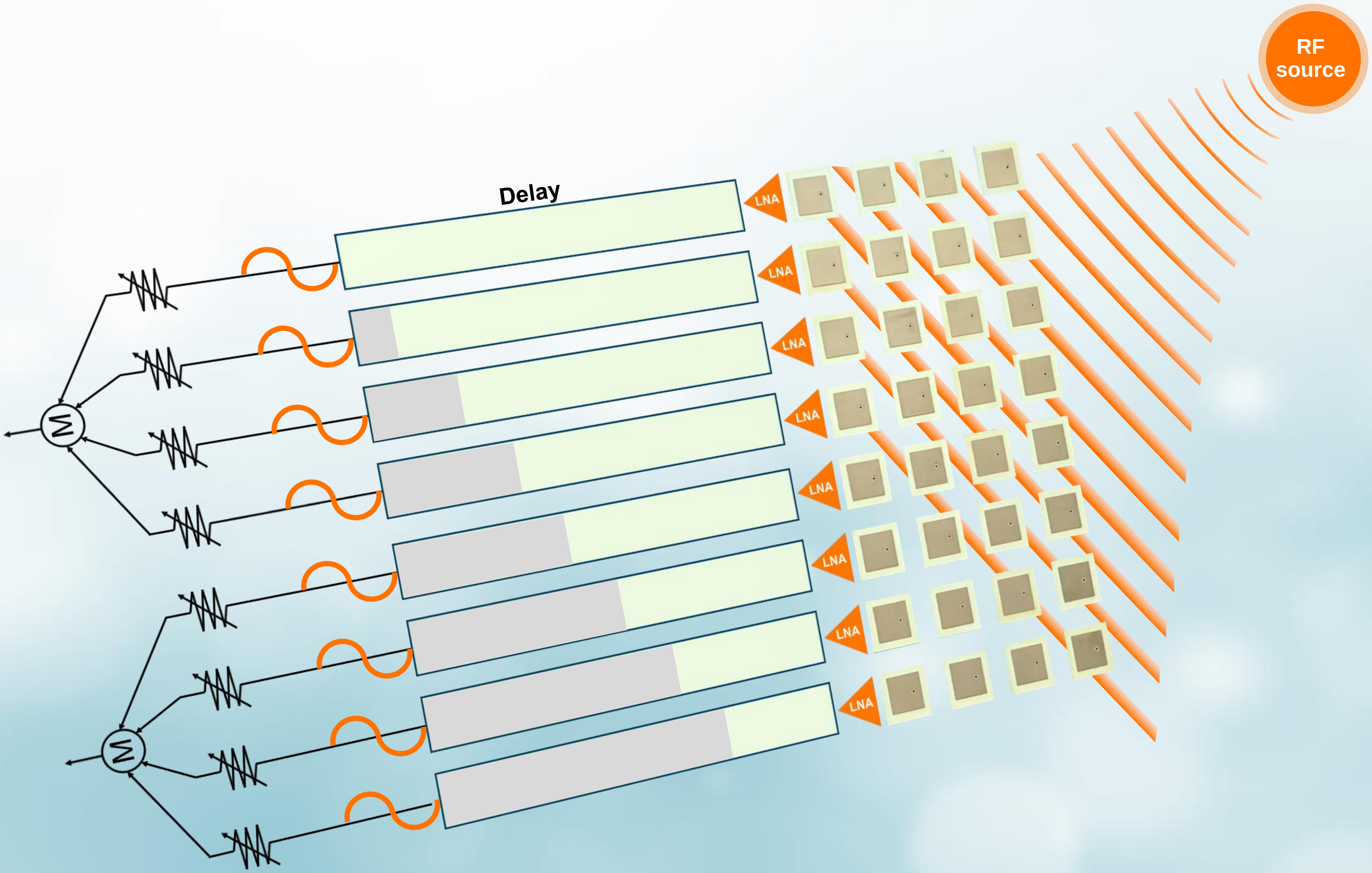

RF
source
Delay
LNA
LNA
LNA
LNA
LNA
LNA
LNA
LNA


RF
source
Delay
LNA
LNA
LNA
LNA
LNA
LNA
LNA
LNA

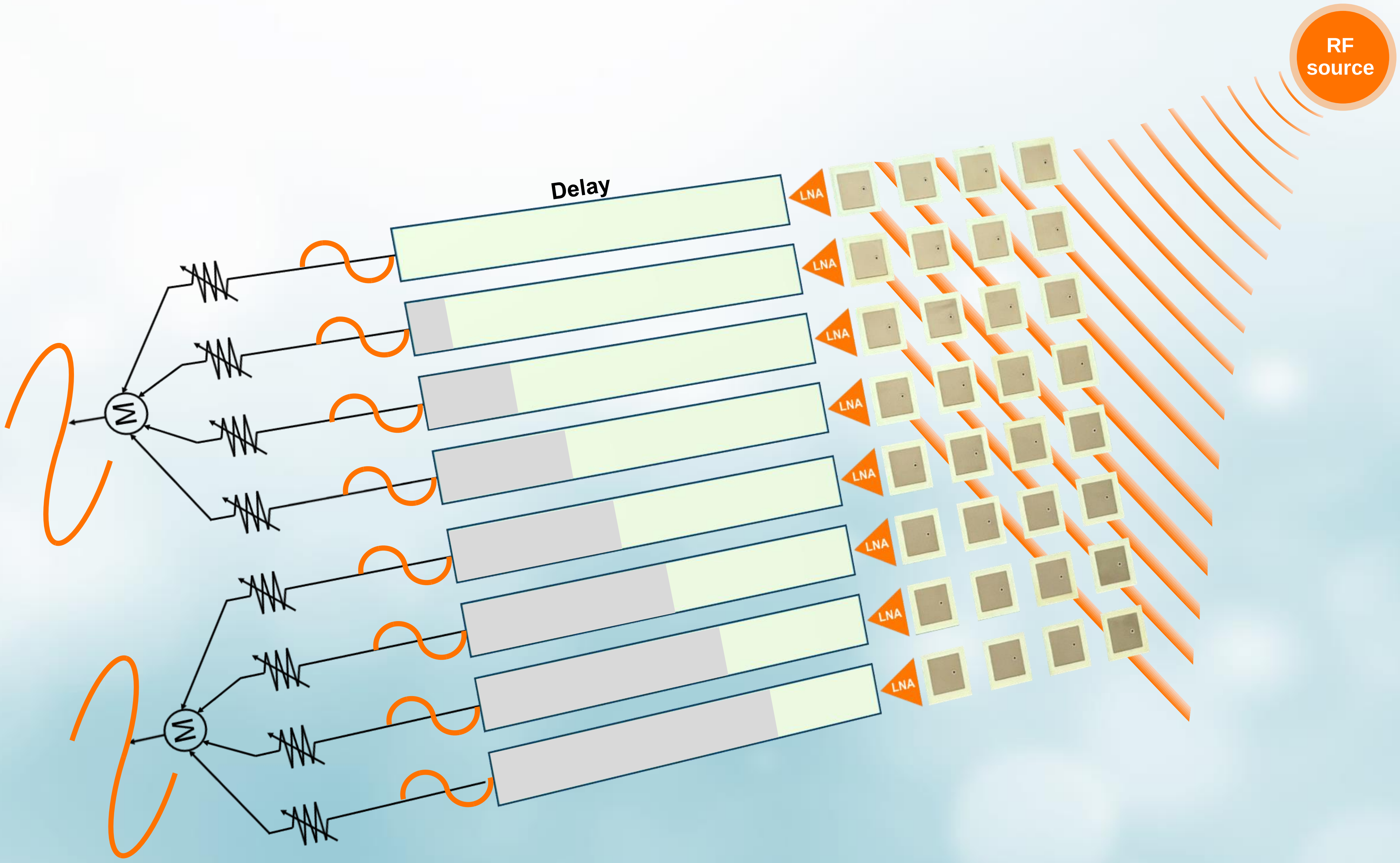

RF source
Delay
LNA
LNA
LNA
LNA
LNA
LNA
LNA
LNA
Σ
Σ

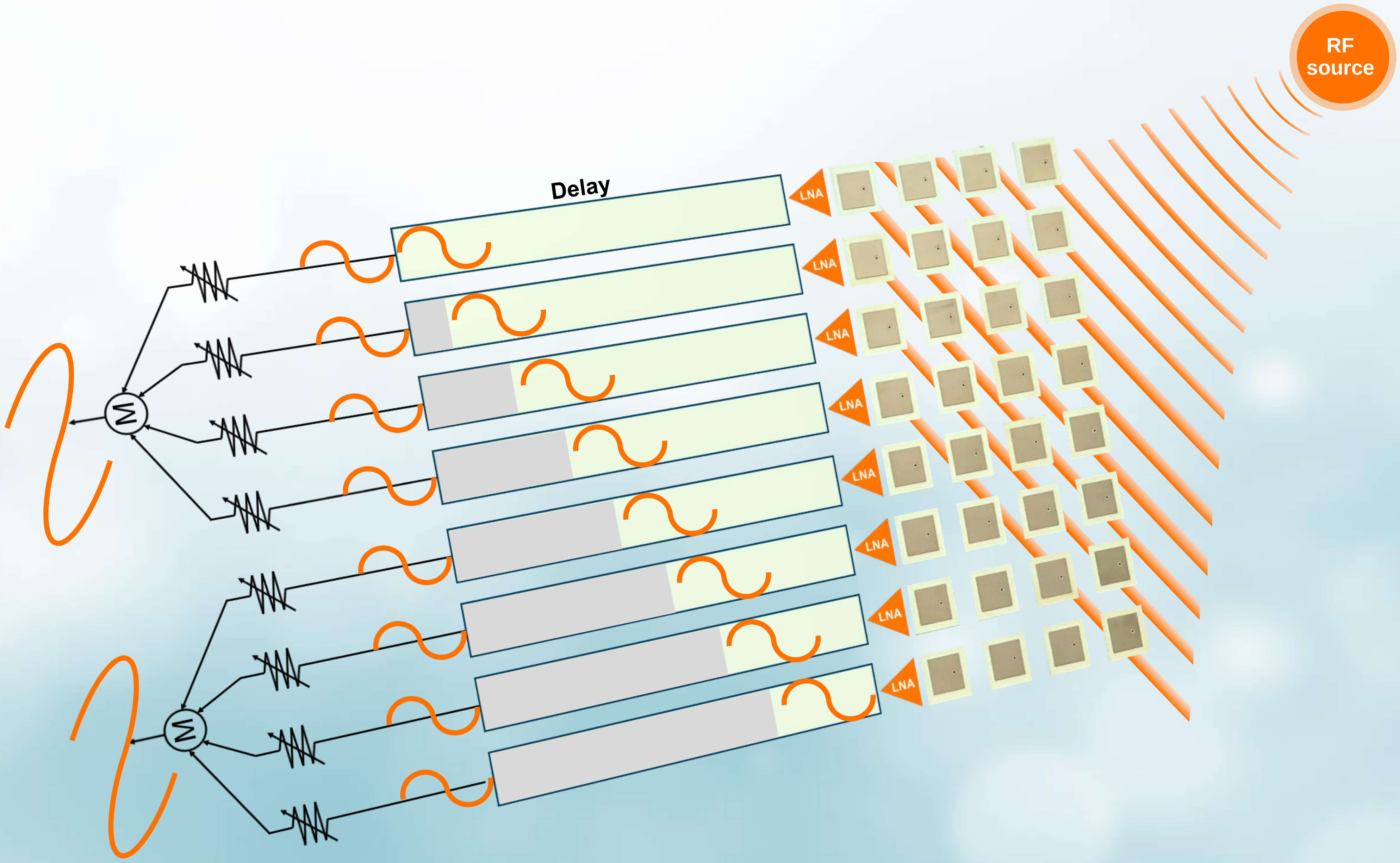


ANALOG DEVICES
AHEAD OF WHAT'S POSSIBLE™

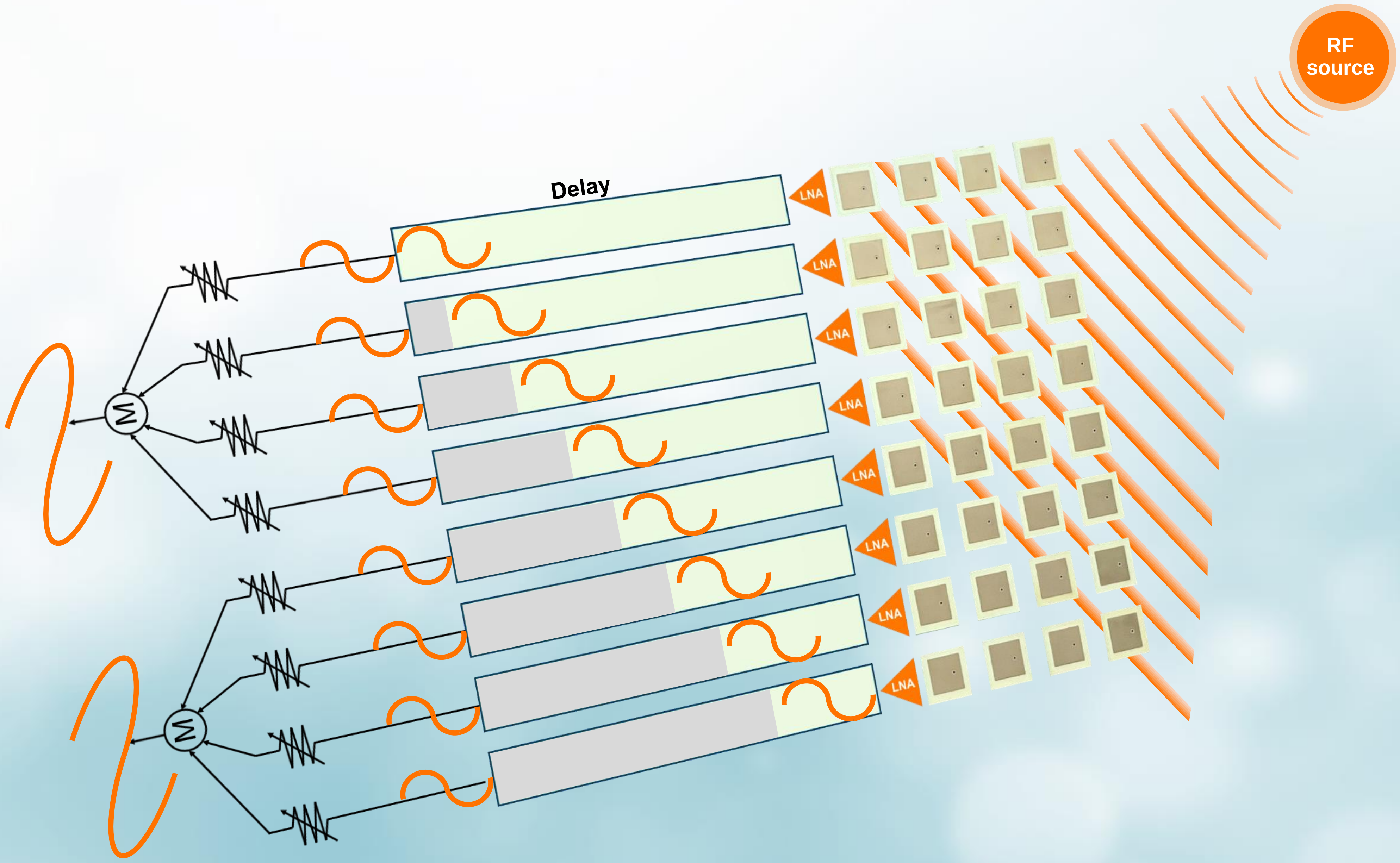


ANALOG DEVICES
AHEAD OF WHAT'S POSSIBLE™


RF source
Delay
LNA
LNA
LNA
LNA
LNA
LNA
LNA
LNA



RF
source
Delay
LNA
LNA
LNA
LNA
LNA
LNA
LNA
LNA

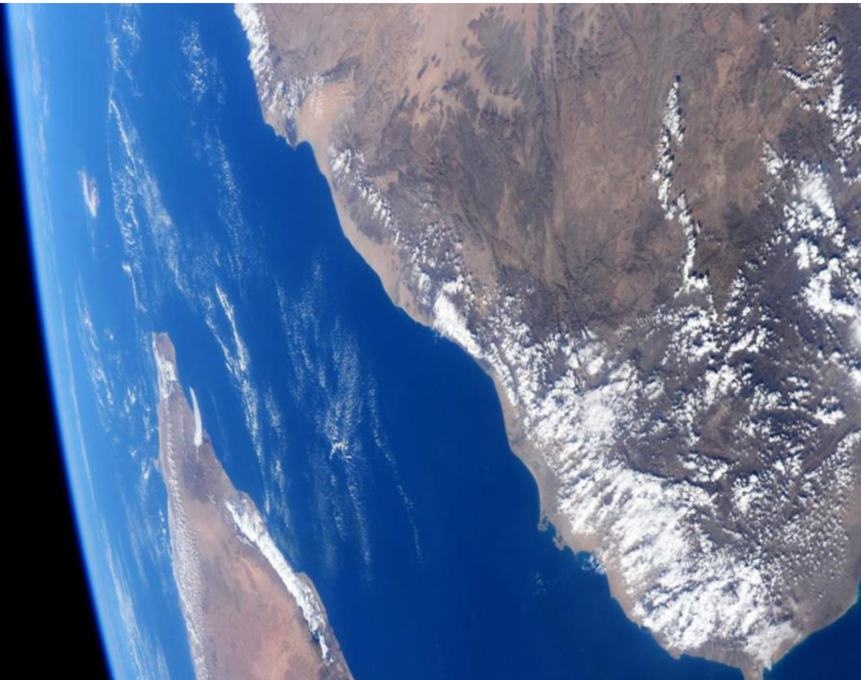

RF source
Delay
LNA
LNA
LNA
LNA
LNA
LNA
LNA
LNA

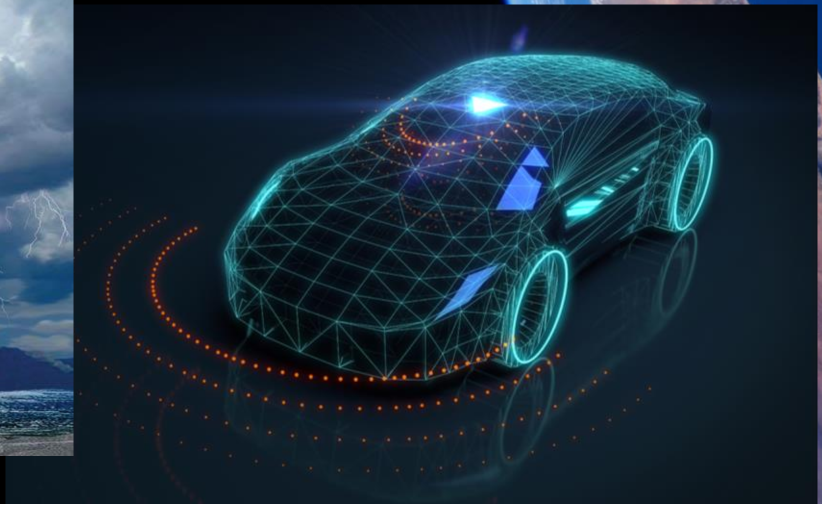
Where is Phased Array Beamforming Used?

Mobile Communications

RADAR

Satellite Communications

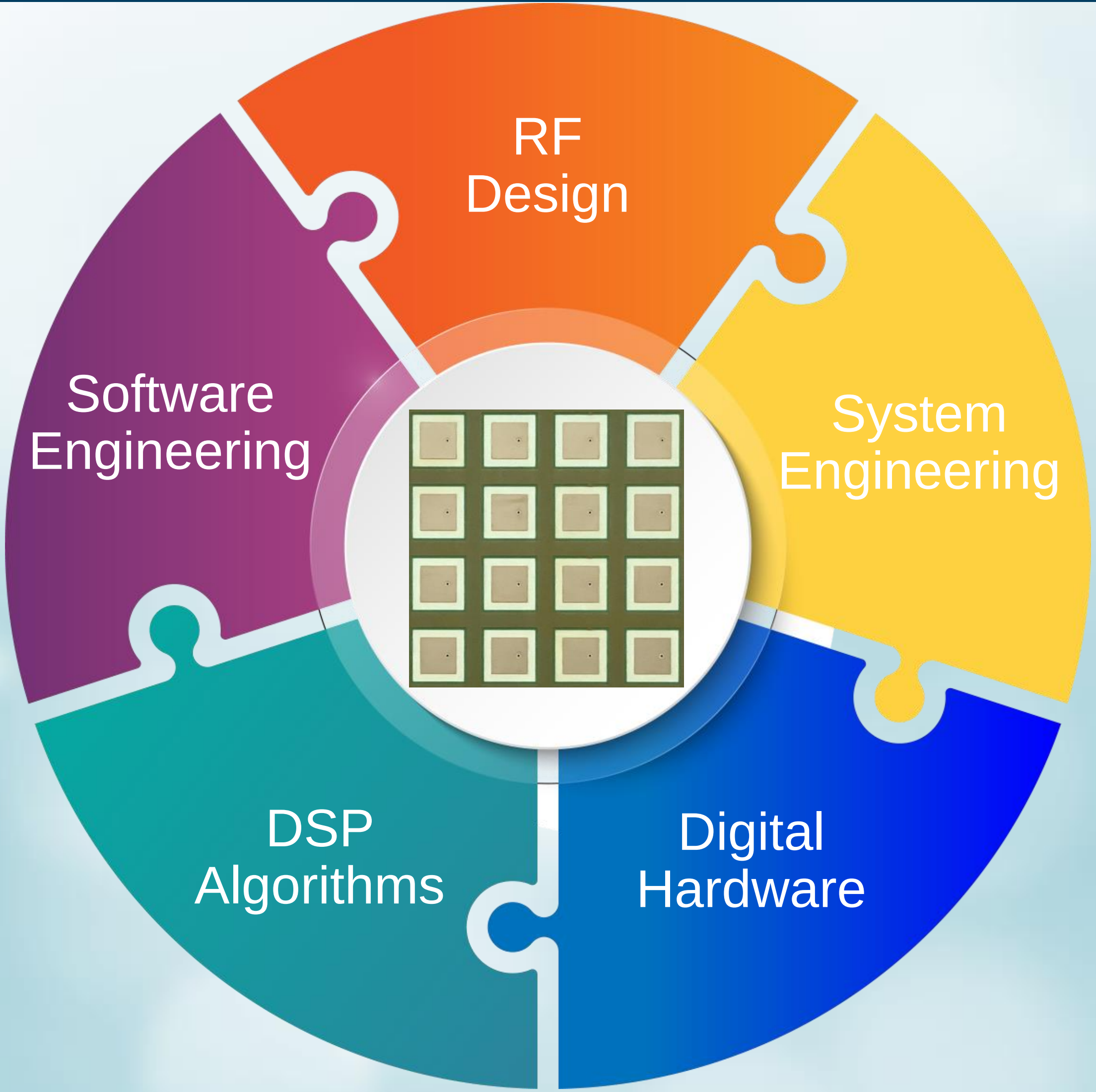




Purpose for the Phased Array Workshop

- Phased Array Radar Development Requires:
  - RF Hardware Design
  - Software Engineering
  - System Design
  - Algorithm Design (comms and radar)
  - HDL Engineering

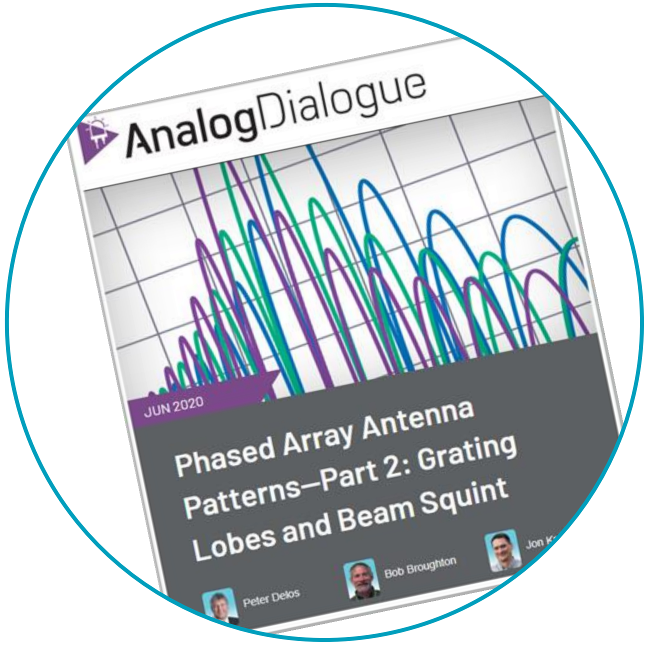
- So with so much entailed, how can we get started?



ANALOG DEVICES
AHEAD OF WHAT'S POSSIBLE™

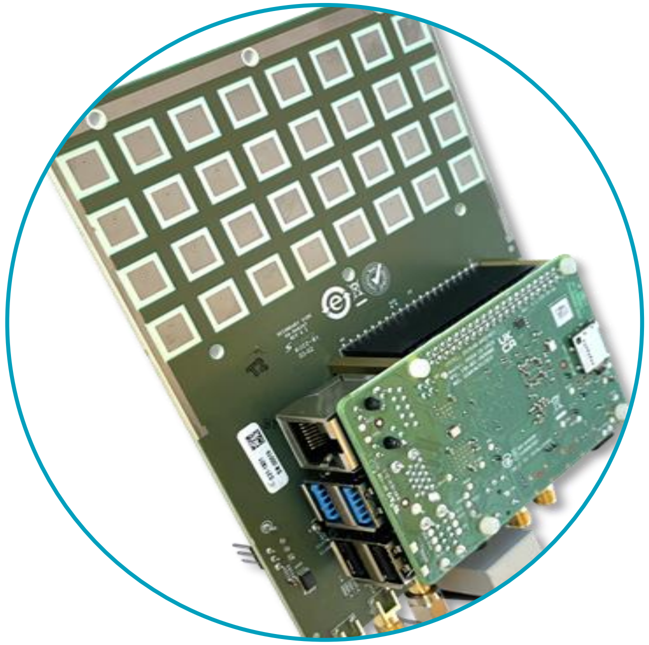
Phased Array Workshop Goals:

1. Gain an **intuitive** understanding of beamforming concepts
2. **Hands on** experimenting with these concepts
3. Path to quickly **prototype** your own phased array system



Math and Theory

+



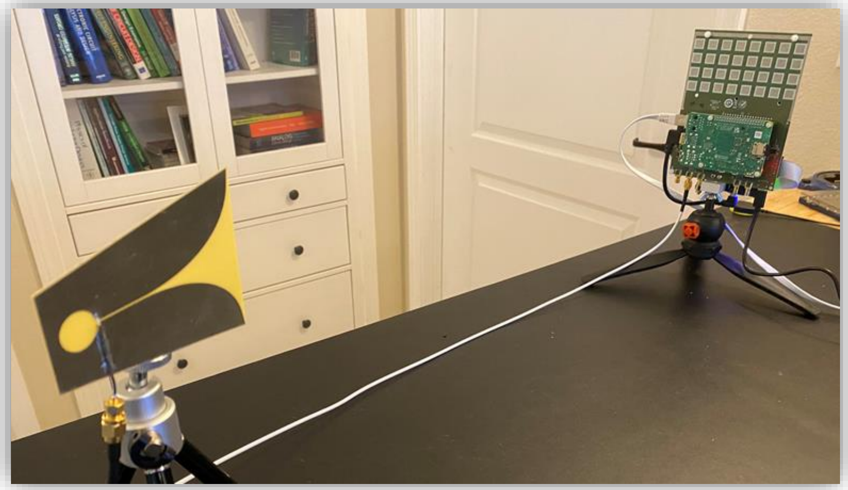
Accessible Hardware

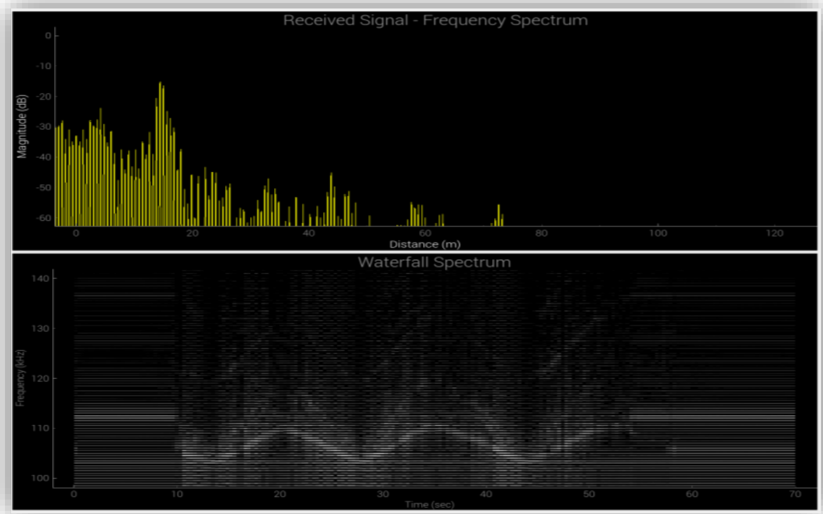
=

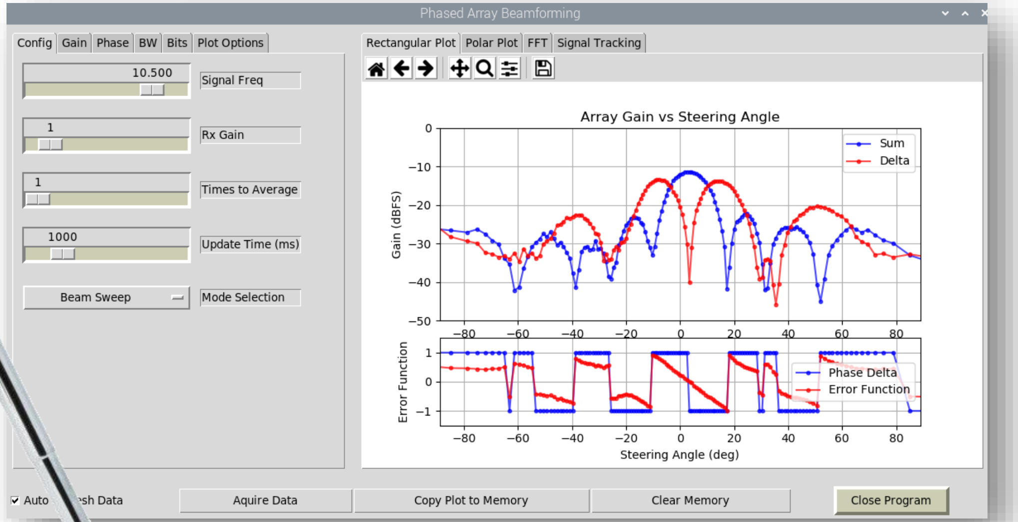
Understanding

Introducing the Phaser

The Phaser: *X Band Phased Array RADAR System*



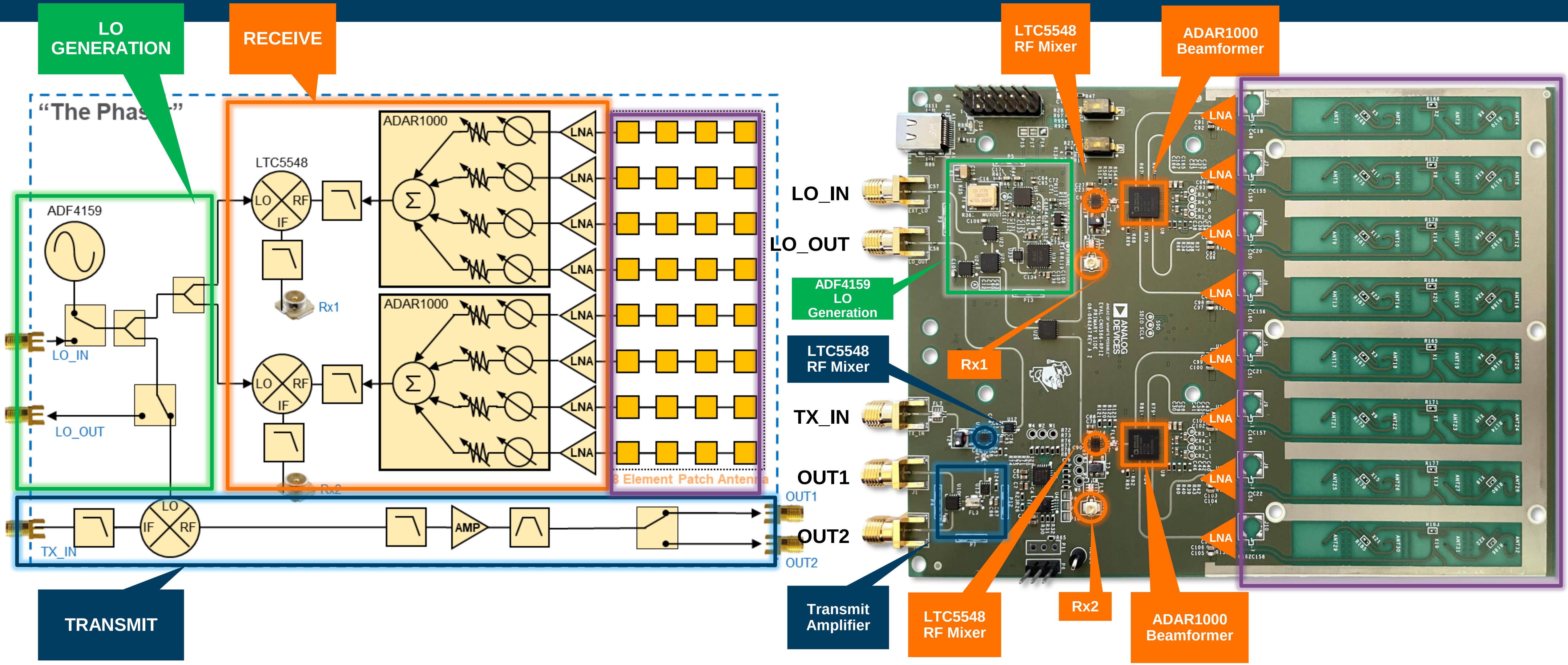




- **Phased Array Education/Prototyping**
- Comms: 10-11 GHz operation
- Radar: 500MHz BW FMCW Chirps
- 8 channel Receive, 2 channel Transmit
- Open source software, hardware
- Price: ~$2500 for the entire kit
- www.analog.com/cn0566
- wiki.analog.com/phaser

ANALOG DEVICES
AHEAD OF WHAT'S POSSIBLE™

Phaser Block Diagram



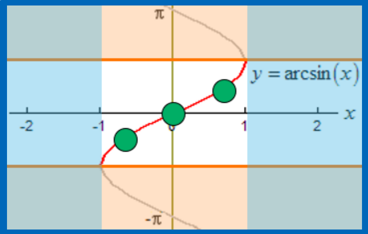
RECEIVE | LO GENERATION | TRANSMIT

ANALOG DEVICES
AHEAD OF WHAT'S POSSIBLE™

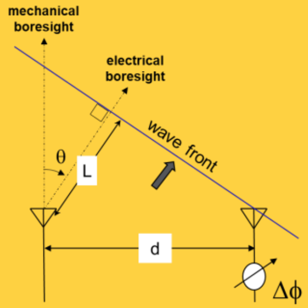
Phased Array Workshop Topics

Digitizer

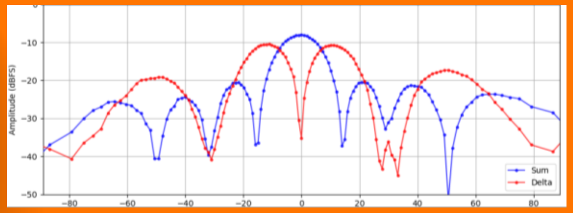
Antenna Impairments



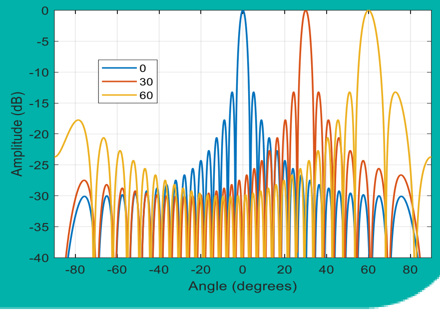
Steering Angle



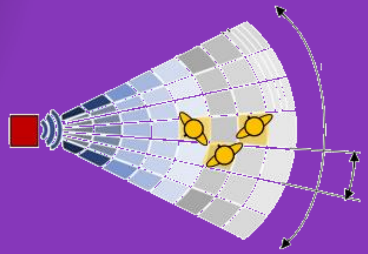
Monopulse Tracking



Antenna Patterns



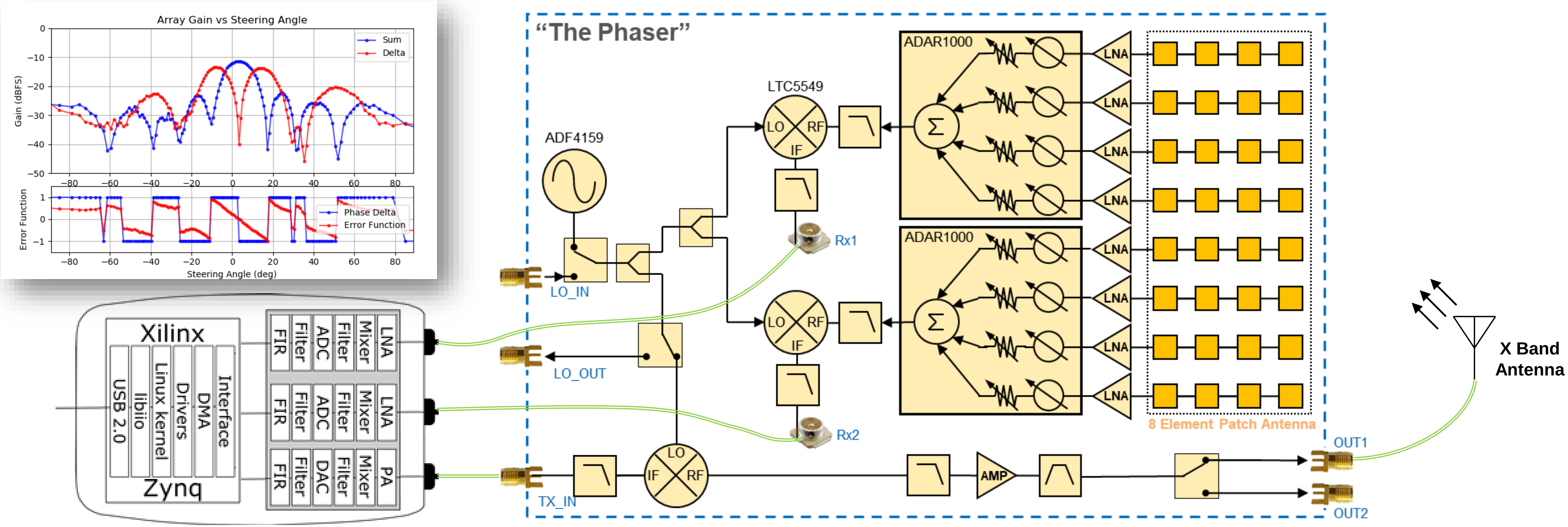
Radar



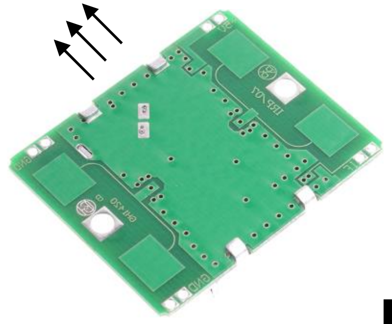
4 September 2023

ANALOG DEVICES
AHEAD OF WHAT'S POSSIBLE™

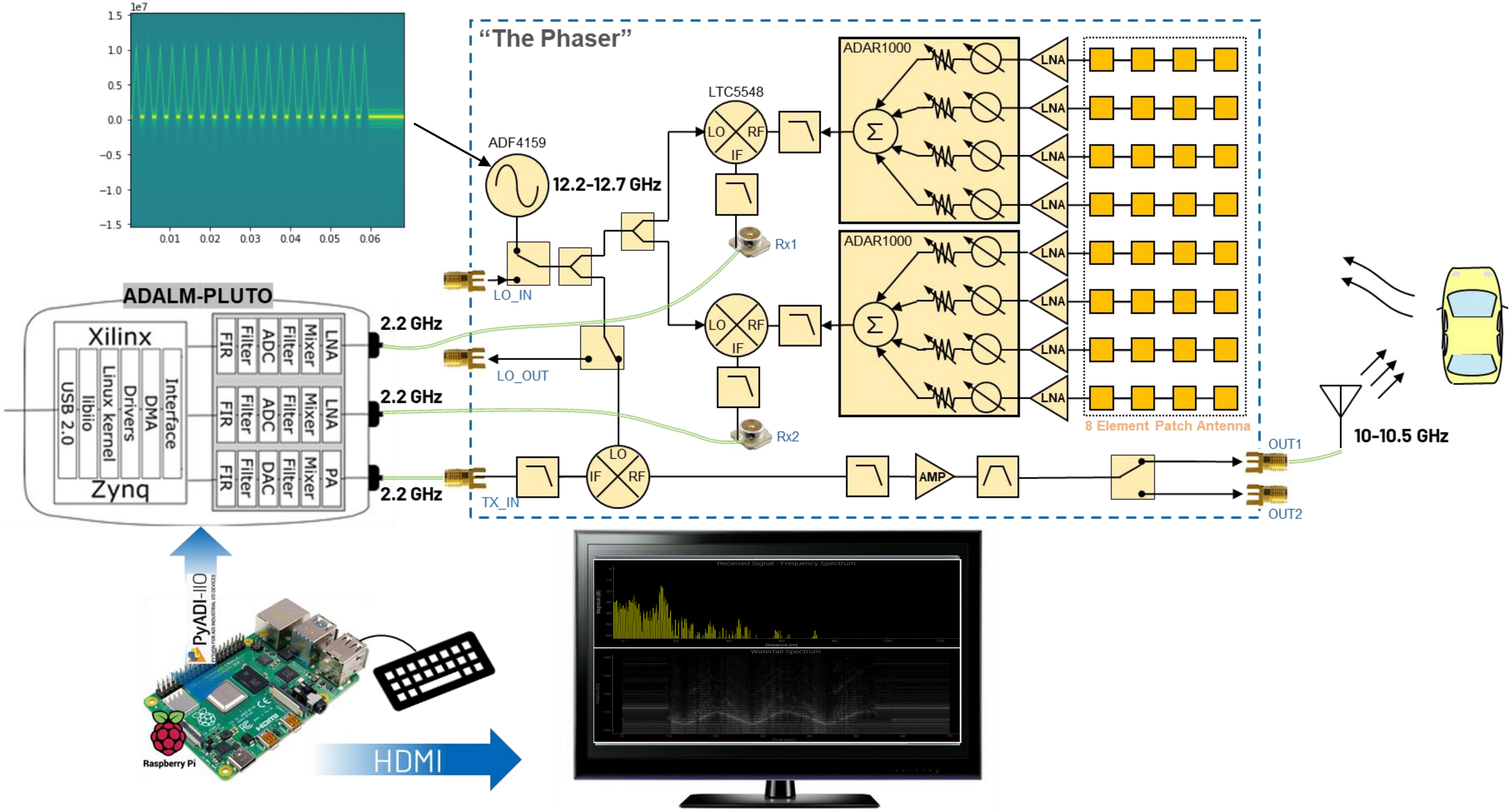
Example: Monopulse Tracking



- Plot antenna pattern with Pluto
- Move the Tx antenna and track elevation angle
- Control Tx freq to observe grating lobes and beam squint
- Or HB100 for battery powered 10.5 GHz Transmitter
  - https://www.digikey.com/htmldatasheets/production/2071176/0/0/1/sen0192.html

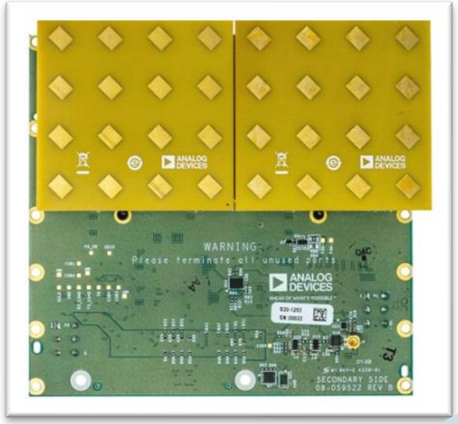


Phaser Basic FMCW Radar Setup

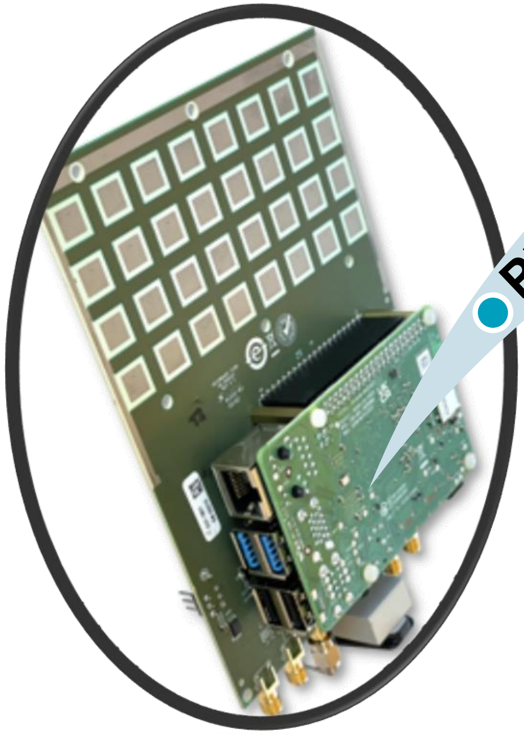


ANALOG DEVICES
AHEAD OF WHAT'S POSSIBLE™

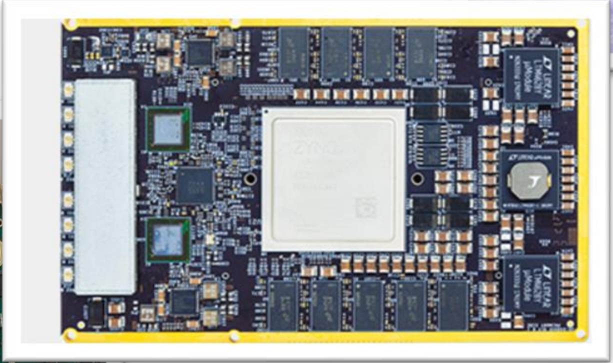
From Phaser → Prototype → Production



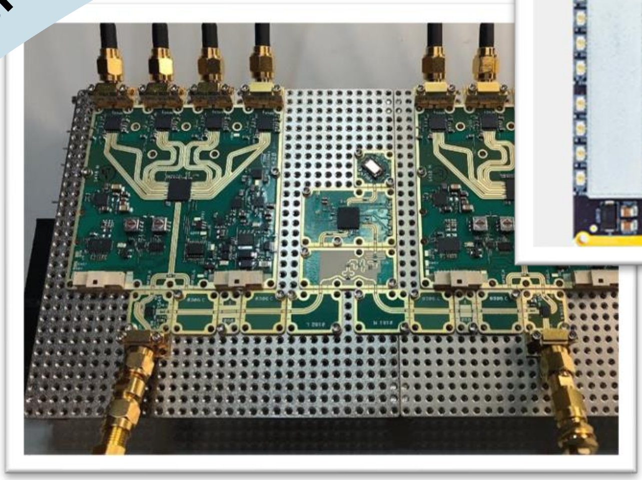
X-Band Phased Array Platform



Final Production



Production Ready SOMs



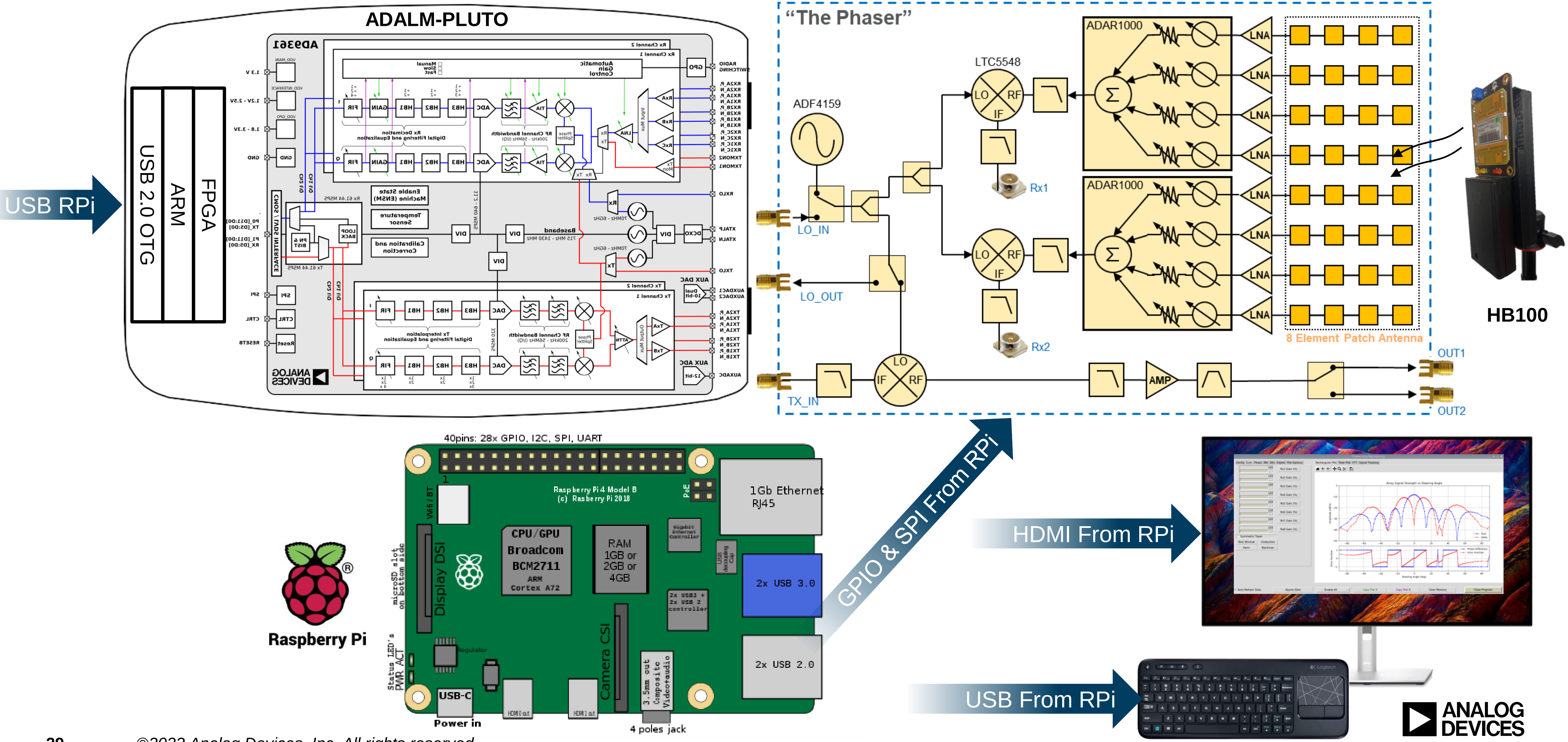
X-Microwave Prototyping System

Same software can be used at each stage!

ANALOG DEVICES
AHEAD OF WHAT'S POSSIBLE™

How to control the Phaser

System Level Diagram



ANALOG DEVICES
AHEAD OF WHAT'S POSSIBLE™

How to Program and Control the ADALM-Phaser

- Three Easy Methods:
  - Matlab → use Analog Devices Board Support Package in Matlab
  - Python → use PyADI-IIO
  - GNU Radio → use PyADI-IIO in the Python Block/Module of GRC
- PyADI-IIO:
  - https://analogdevicesinc.github.io/pyadi-iio/
  - PyADI-IIO is a python abstraction module for ADI hardware with IIO drivers to make them easier to use."
  - "glue layer" between IIO (which has a bit of a learning curve) and doing something useful
  - Pre-installed on ADI Kuiper Linux (ADI's custom Raspberry Pi OS, with device drivers and utilities)
  - Under the Surface: Built on the industry-standard Linux Industrial I/O framework:
    - Cross platform API (Windows/Linux/Mac)
    - Multiple bindings (Python, MATLAB, C, C#)

PYADI-IIO makes Python Control Easy!

PyADI-IIO
PYTHON FOR ADI INDUSTRIAL I/O DEVICES

- How easy is PYADI-IIO????
- Grab a chunk of data from Pluto a few lines of code:

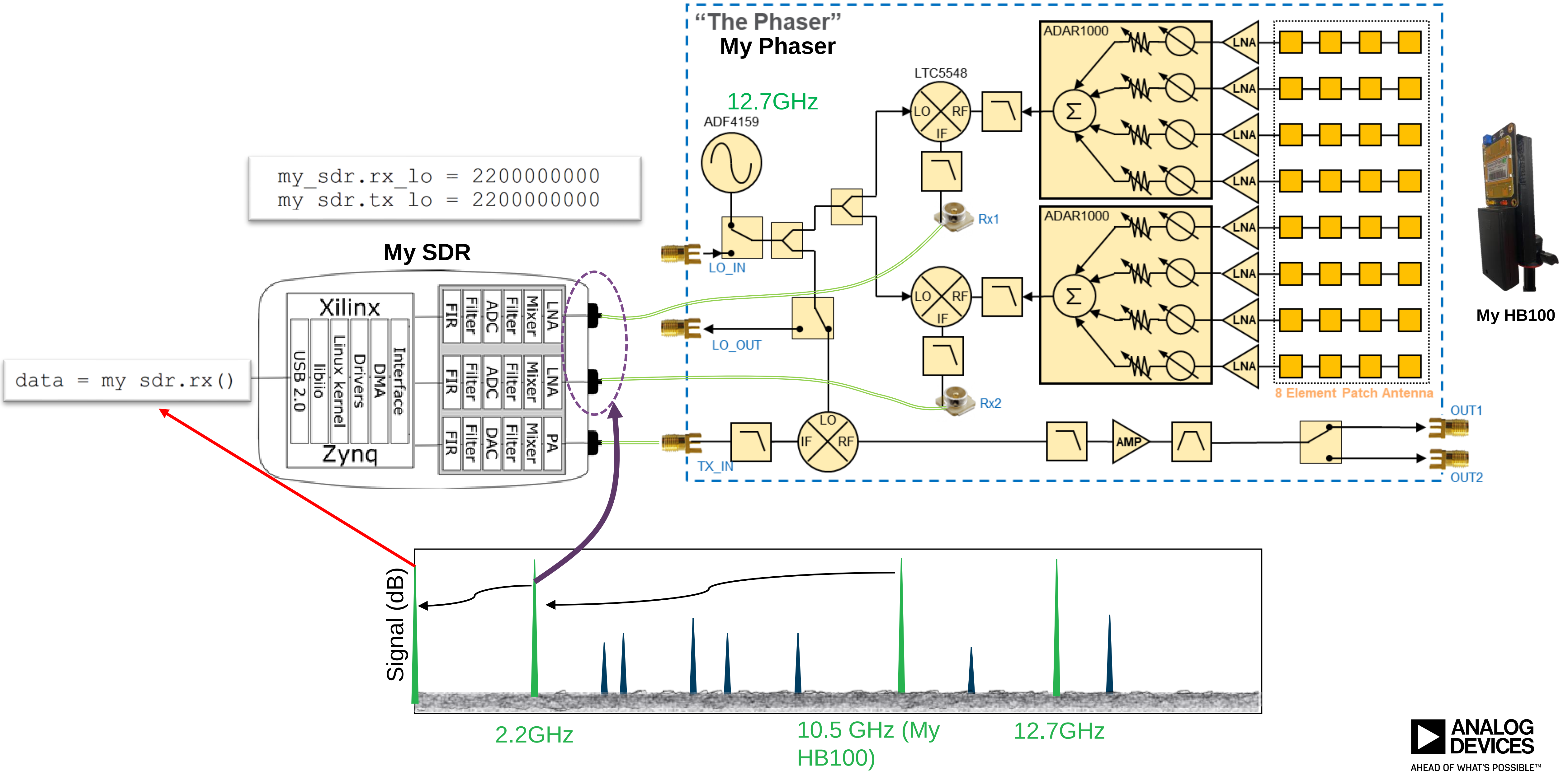
```
import adi

# Create radio
my_sdr = adi.Pluto()

# Configure properties
my_sdr.rx_lo = 2200000000
my_sdr.tx_lo = 2200000000

# Collect data
data = my_sdr.rx()
```

- Full example script here:

https://github.com/analogdevicesinc/pyadi-iio/blob/master/examples/pluto.py

ANALOG DEVICES
AHEAD OF WHAT'S POSSIBLE™

What can we do with Python and Pyadi-iio?



What does the Python look like to Control Phaser?

Raspberry Pi's IP address

Pluto object we created earlier

```
import adi

# Create Phaser object
my_phaser = adi.cn0566(uri="ip:phaser.local", rx_dev=my_sdr)

# Set all ADAR1000 channels to phase = 0 deg
# and apply a Blackman taper to the array
gain_list = [8, 34, 84, 127, 127, 84, 34, 8]  #Blackman taper
for i in range(0, 8):
    my_phaser.set_chan_phase(i, 0)
    my_phaser.set_chan_gain(i, gain_list[i])
```

- Full example script here:

https://github.com/analogdevicesinc/pyadi-iio/blob/cn0566_dev/examples/cn0566/cn0566_minimal_example.py

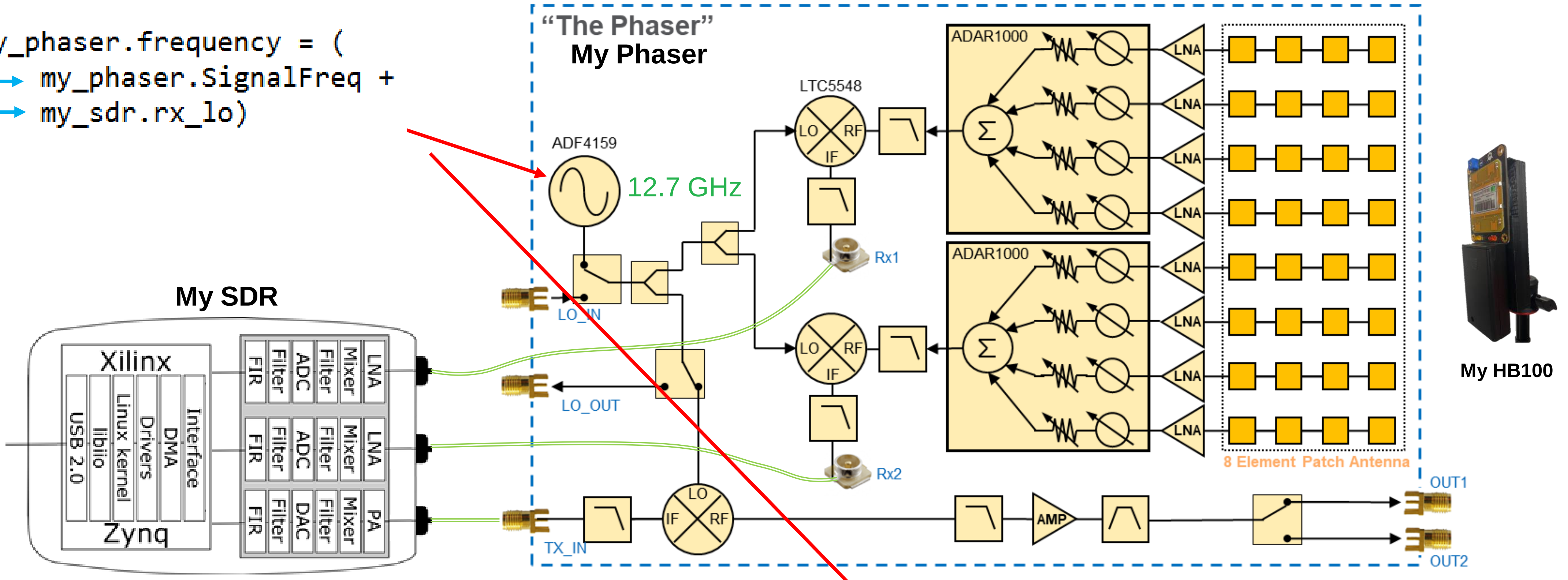
ANALOG DEVICES
AHEAD OF WHAT'S POSSIBLE™

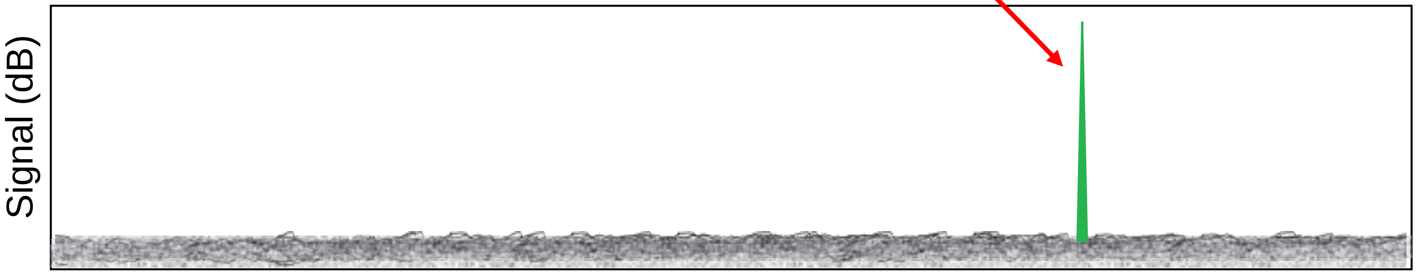
What can we do with Python and Pyadi-iio?

```
my_phaser.frequency = (
    my_phaser.SignalFreq +
    my_sdr.rx_lo)
```

10.5 GHz → my_phaser.SignalFreq

2.2 GHz → my_sdr.rx_lo



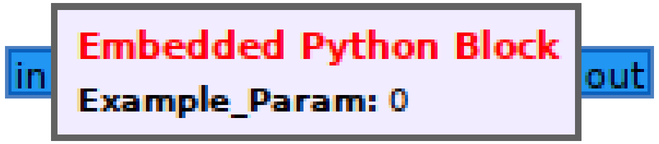


12.7 GHz
(LO)

GNU Radio and Phaser

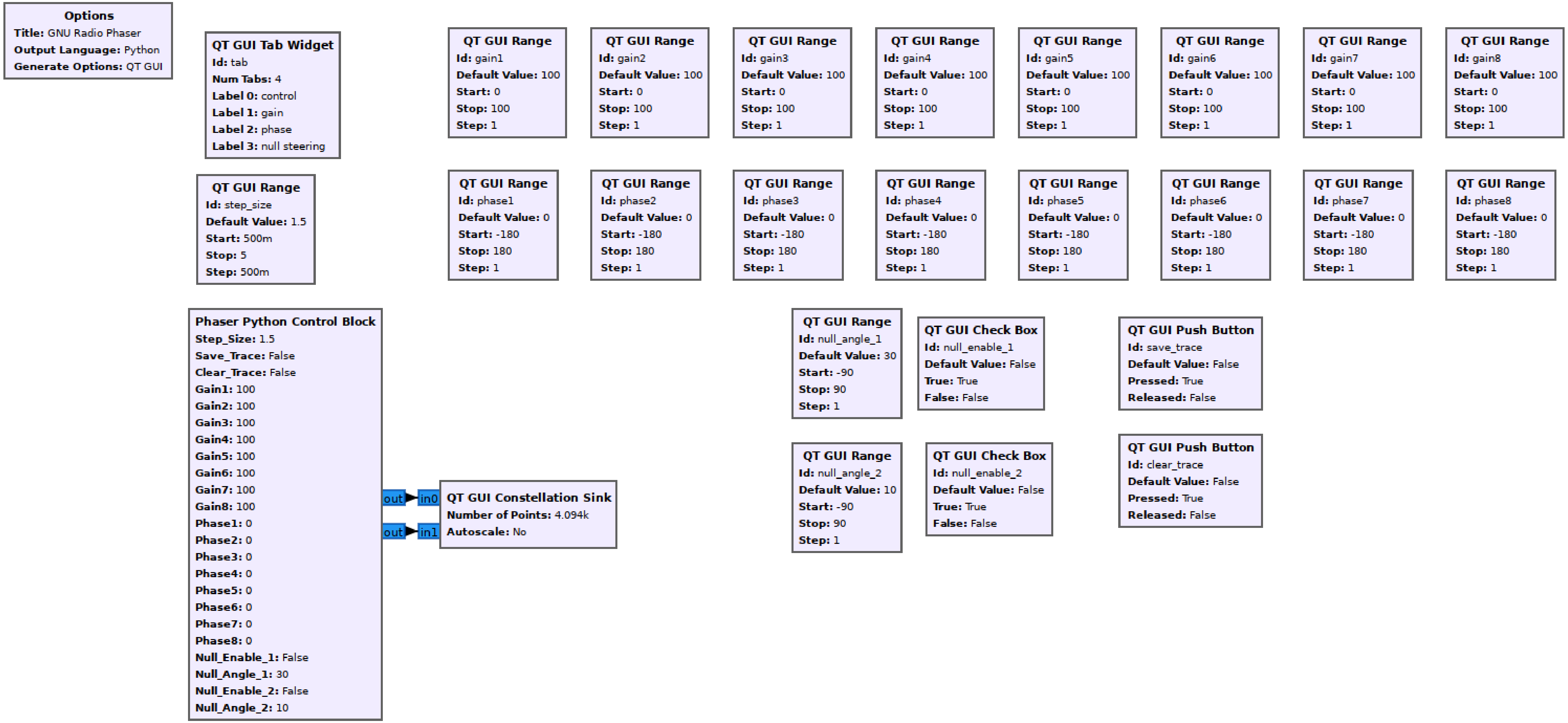
Running the Phaser in GNU Radio Companion

- **Why GNU Radio?**
  1. Easy to create a GUI, without burdening the Python code
  2. Very fast GUI updates
- **Why Not Build an OOT** (out of tree) module for GRC?
  1. I don’t know how to do that…..
  2. They always seem to fall out of step with the GRC releases
  3. Using Python block allows us to easily reuse all of our existing Python code examples

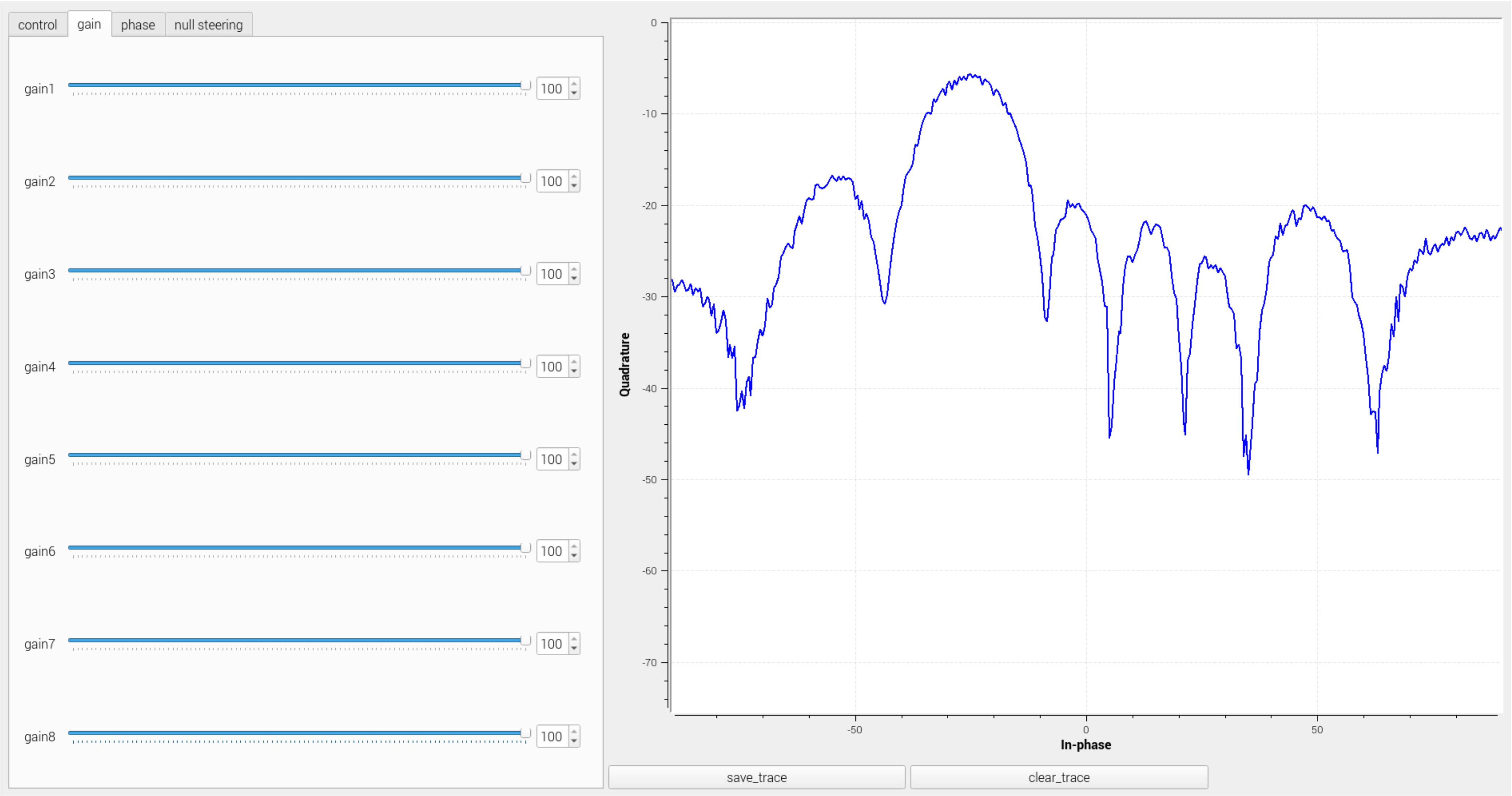


- **Why are we using GRC 3.8?** Instead of 3.10 or 3.11?
  1. “sudo apt get install” only gives us GRC 3.8
  2. This all will work, without any changes to later versions
  3. Later versions will also give us more plotting and GUI options—which would be nice

GRC Flowgraph:

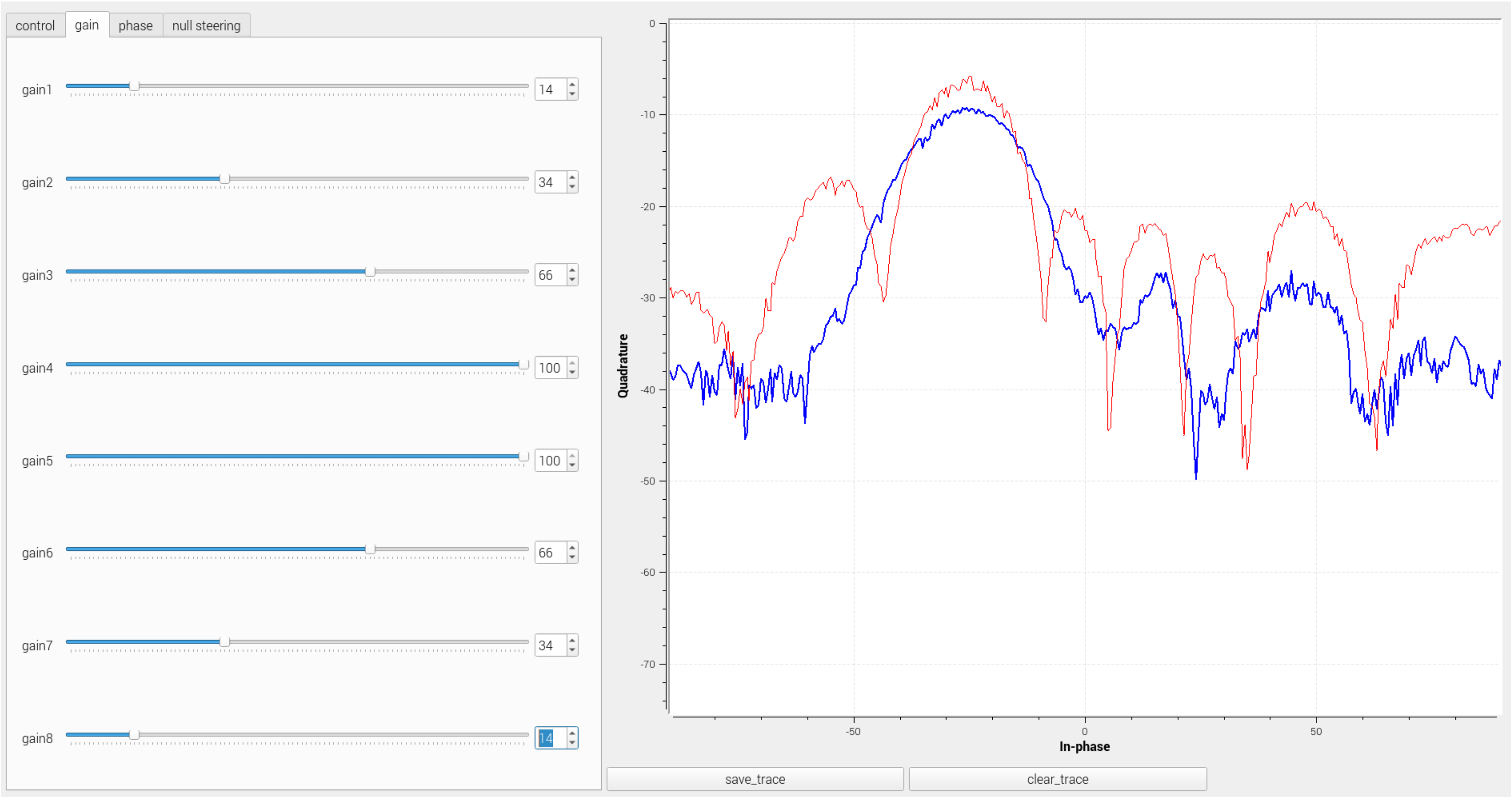


GRC GUI *<<<LIVE DEMO>>>*



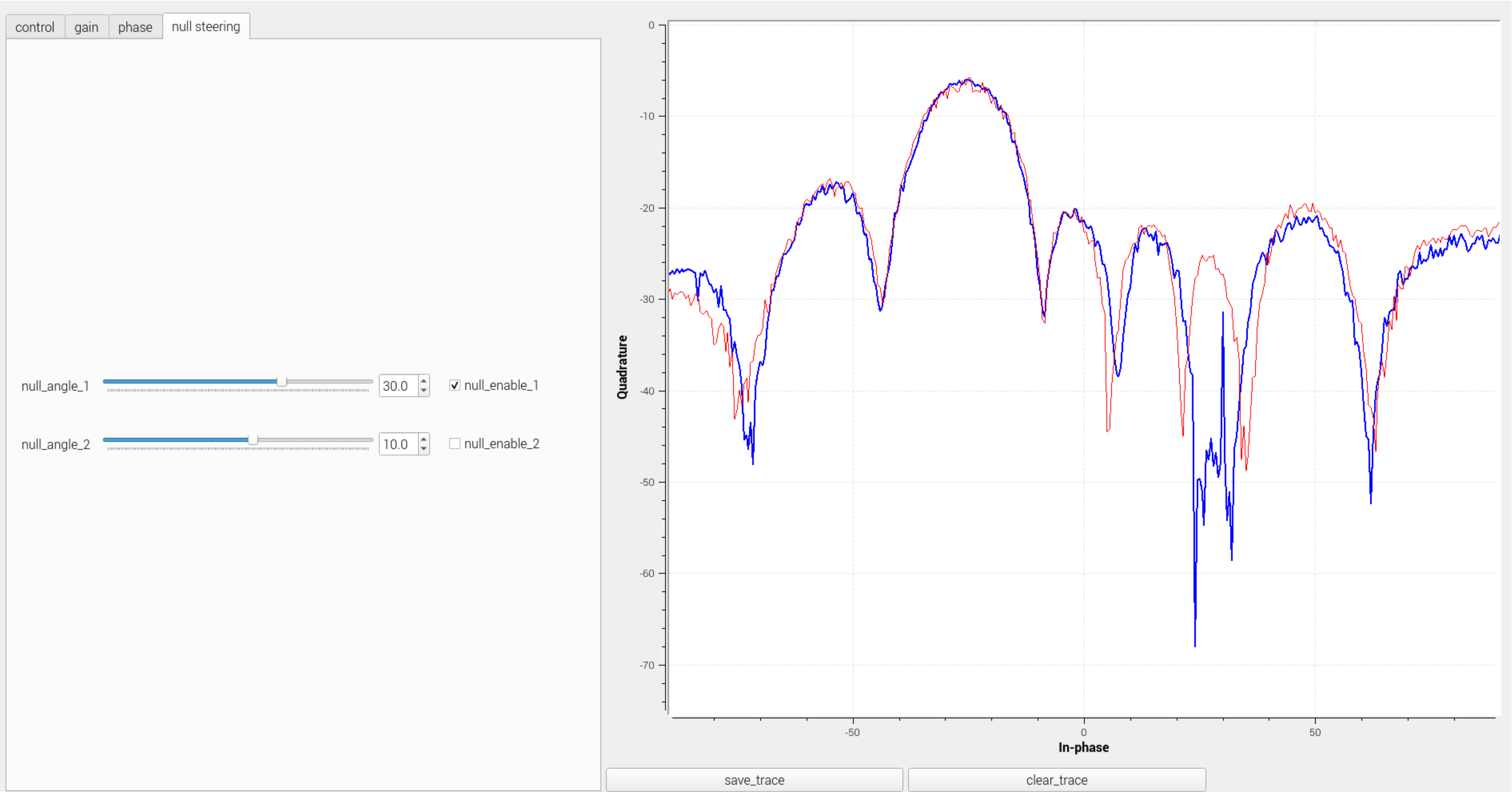
ANALOG DEVICES
AHEAD OF WHAT'S POSSIBLE™

Beam Tapering Example: *<<<LIVE DEMO>>>*



ANALOG DEVICES
AHEAD OF WHAT'S POSSIBLE™

Null Steering Example: *<<<LIVE DEMO>>>*



ANALOG DEVICES
AHEAD OF WHAT'S POSSIBLE™

Code Walkthrough:

1. Link GRC GUI input objects to the Python block:

```
class blk(gr.sync_block):
    def __init__(self, step_size=3, save_trace=False, clear_trace=False,
                 gain1=100, gain2=100, gain3=100, gain4=100, gain5=100, gain6=100, gain7=100, gain8=100,
                 phase1=0, phase2=0, phase3=0, phase4=0, phase5=0, phase6=0, phase7=0, phase8=0,
                 null_enable_1 = False, null_angle_1=0, null_enable_2 = False, null_angle_2=0):  # only
        """arguments to this function show up as parameters in GRC"""
        gr.sync_block.__init__(
            self,
            name='Phaser Python Control Block',   # will show up in GRC
            in_sig=[],
            out_sig=[np.complex64, np.complex64]
        )
        # if an attribute with the same name as a parameter is found,
        # a callback is registered (properties work, too)

        # ==============================================================================
        # User parameters
        # ==============================================================================
        self.step_size = step_size  # steering angle step size (in degrees)
        self.save_trace = save_trace
        self.clear_trace = clear_trace
        self.saved_trace = np.ones(4094)*(0-100000j)

        self.gain1=gain1
        self.gain2=gain2
        self.gain3=gain3
        self.gain4=gain4
        self.gain5=gain5
        self.gain6=gain6
        self.gain7=gain7
        self.gain8=gain8
```

Code Walkthrough:

2. Create PYADI-IIO objects for Pluto and the ADAR1000 array:

```
while attempt:
    try:
        # Initialize Pluto
        self.sdr = SDR.SDR_init(
            sdr_address,
            SampleRate,
            Tx_freq,
            Rx_freq,
            Rx_gain,
            Tx_gain,
            config.buffer_size,
            )
        SDR.SDR_LO_init(rpi_ip, LO_freq)  # Set Phaser's ADF4159 to the LO_freq

        # Intialize the ADAR1000 receive array
        self.rx_array = adi.adar1000_array(
            uri=rpi_ip,
            chip_ids=["BEAM0", "BEAM1"],  # these are the ADAR1000s' labels in the device tree
            device_map=[[1], [2]],
            element_map=[[1, 2, 3, 4, 5, 6, 7, 8]],
            device_element_map={
                1: [7, 8, 5, 6],  # i.e. channel2 of device1 (BEAM0), maps to element 8
                2: [3, 4, 1, 2],
                },
            )
        attempt = False
        print('Connected to Phaser')
    except:
        print('Could not connect to Phaser')
        continue
```

Code Walkthrough:

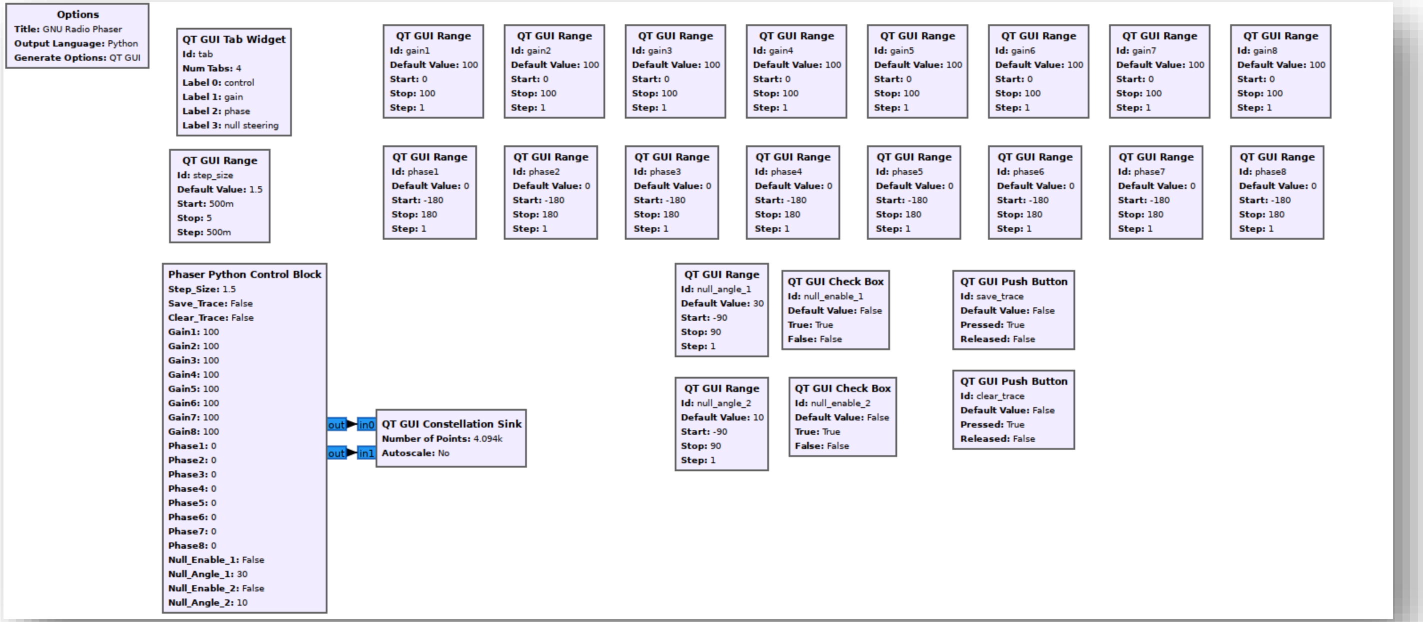
1. Do work! Phase shifting and Null Steering:

```
def work(self, input_items, output_items):
    save_this_trace=False
    clear_this_trace=False
    enable_null_1 = self.null_enable_1
    enable_null_2 = self.null_enable_2
    if enable_null_1 == True:
        polar_null_phases = self.null_vec(self.null_angle_1)
        wn1 = self.gainList * polar_null_phases   # beam weights for null direction
    if enable_null_2 == True:
        polar_null_phases = self.null_vec(self.null_angle_2)
        wn2 = self.gainList * polar_null_phases   # beam weights for null direction

    SteerValues = np.arange(-90, 90 + self.step_size, self.step_size)
    # Phase delta = 2*Pi*d*sin(theta)/lambda = 2*Pi*d*sin(theta)*f/c
    PhaseValues = np.degrees(
        2*np.pi*self.d* np.sin(np.radians(SteerValues))
        * self.SignalFreq / self.c)
    self.updateGainPhase()

    for x in range(len(PhaseValues)):
        if self.save_trace == True:
            save_this_trace = True
        if self.clear_trace == True:
            clear_this_trace = True
        PhDelta = PhaseValues[x]
        steer_phases = (np.array([0,1,2,3,4,5,6,7])*PhDelta) % 360
        wd = self.gainList * np.exp(1j * np.deg2rad(steer_phases))
        # wd is the beam weights for desired steering direction
        # details here:  https://www.mathworks.com/help/phased/ug/array-pattern-synthesis.html

        if enable_null_1 == True:
            wn1_herm = np.conjugate(wn1.reshape(1,len(wn1)))
            rn = wn1_herm @ wd / (wn1_herm @ wn1)
            wd = wd - wn1 * rn
```

Conclusion

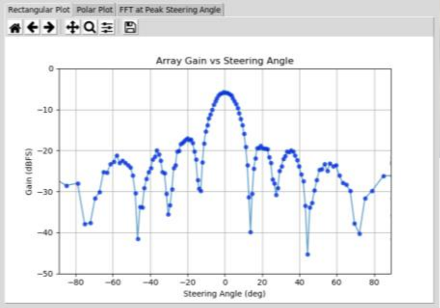
Summary

- What is “Beamforming” and where is it used?
- Introducing the Phaser X Band Phased Array Kit
- How to Control the Phaser with **Python**
- How to Control the Phaser with **GNU Radio Companion**



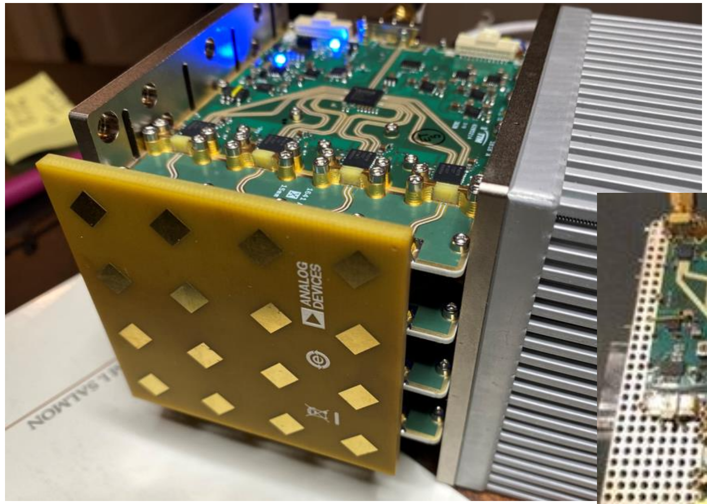
ANALOG DEVICES
AHEAD OF WHAT'S POSSIBLE™

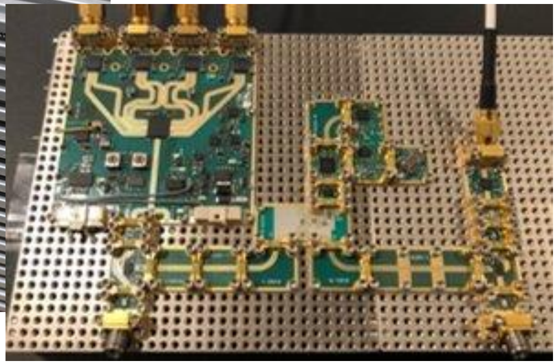
Graduate to All of ADI’s Phased Array Comms and Radar Solutions



- Phaser:
  - Purpose: **Learn** phased array radar and comms
  - Cost: ~$2500
  - Receive only array

- X Microwave:
  - Purpose: Customizable prototyping with near final product performance
  - Cost: $5k-20k
  - Flexibility: **Highest**





- Developer Kits:
  - Purpose:
    - **Closest to real world final solution.**
    - Demonstrate schematic and layout to achieve large scale solution
  - Cost: $30k+
  - Flexibility: **Medium**