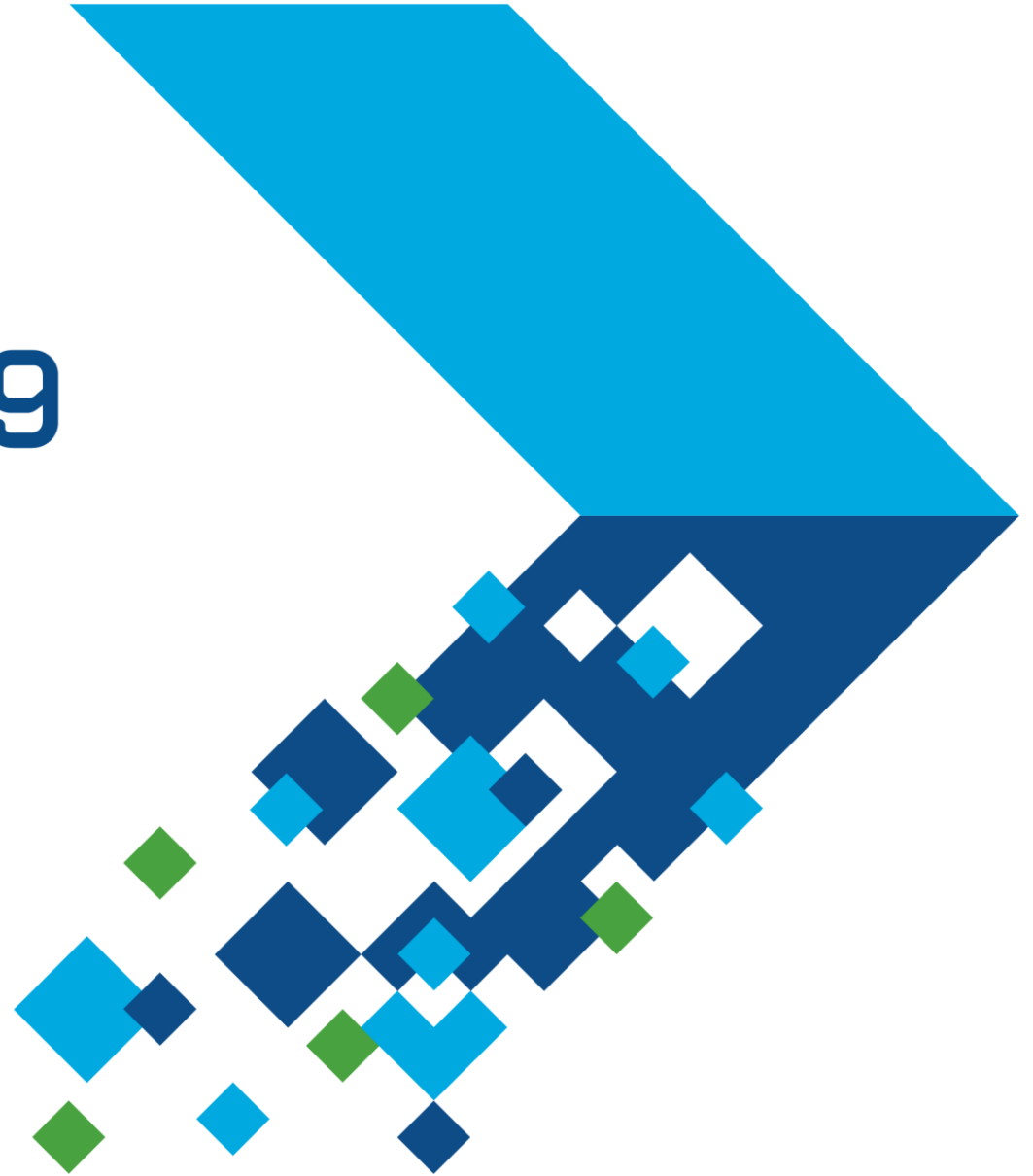


MATLAB EXPO 2019

RF Design and Test Using MATLAB and NI Tools

Tim Reeves – treeves@mathworks.com

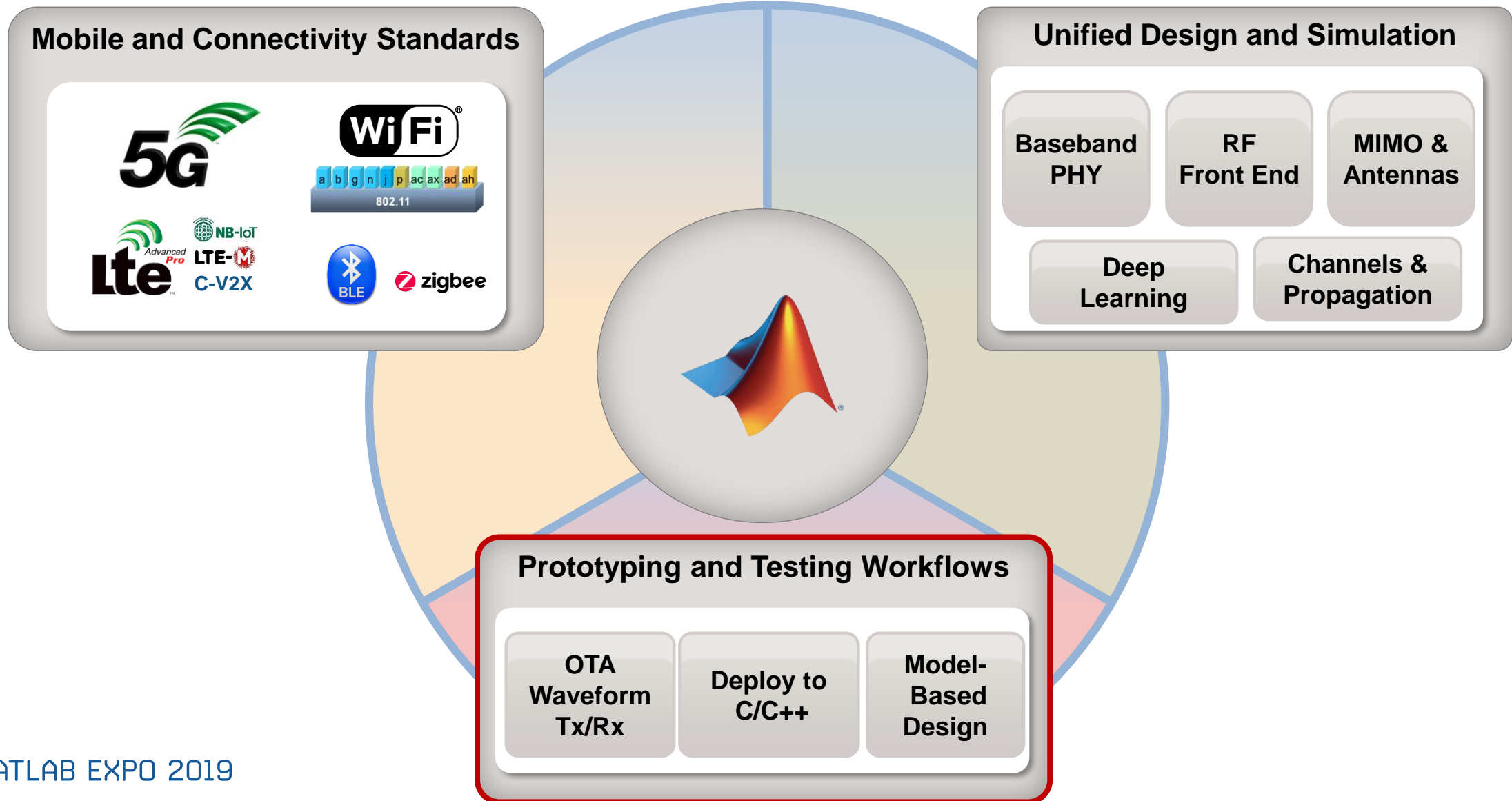
Chen Chang - chen.chang@ni.com



What are we going to talk about?

- How MATLAB and Simulink can be used in a wireless system design workflow
- Wireless Scenario Simulation
- End-to-end Simulation of mmWave Communication Systems with Hybrid Beamforming
- Developing Power Amplifier models and DPD algorithms in MATLAB
- Use of National Instruments PXI for PA characterization with DPD

Common Platform for 5G Development

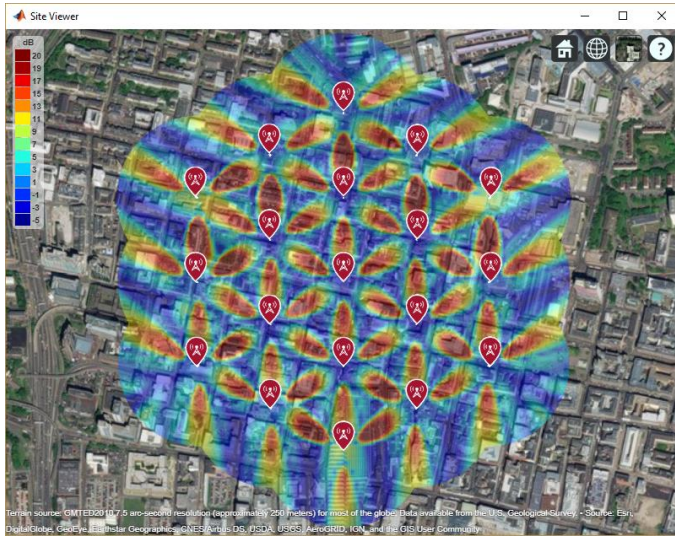


What differentiates high data rate 5G systems from previous wireless system iterations?

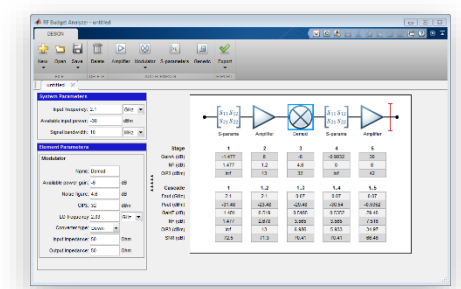
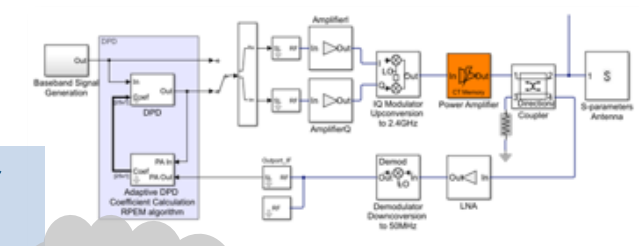
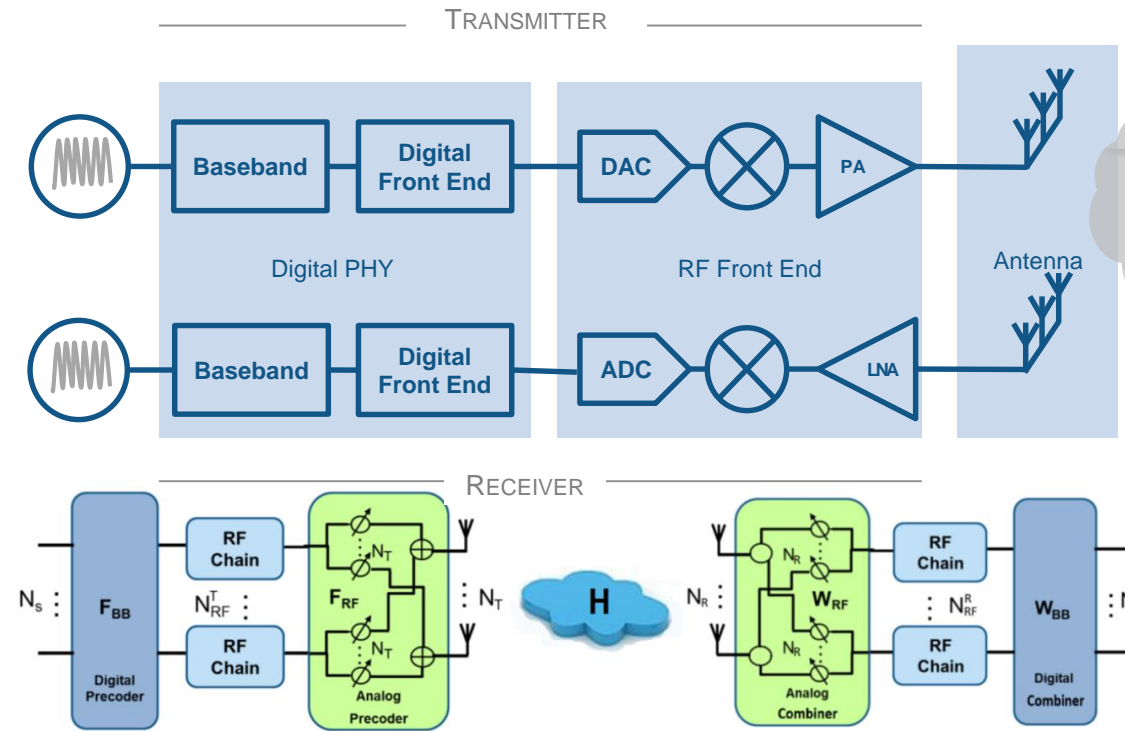
- High data rates (>1 Gbps) requires use of previously “under-used” (mmWave) frequency bands
- mmWave requires MIMO architectures to achieve same performance as sub-6GHz
 - Lower device power and high channel attenuation
- Antenna array, RF, and digital signal processing cannot be designed separately!
 - Large communication bandwidth → digital signal processing is challenging
 - High-throughput DSP → linearity requirements imposed over large bandwidth
 - Wavelength ~ 1mm → small devices, many antennas packed in small areas

How is the presentation set up?

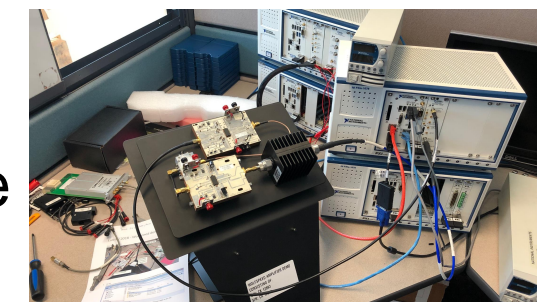
Scenario Modeling



Link Level Modeling

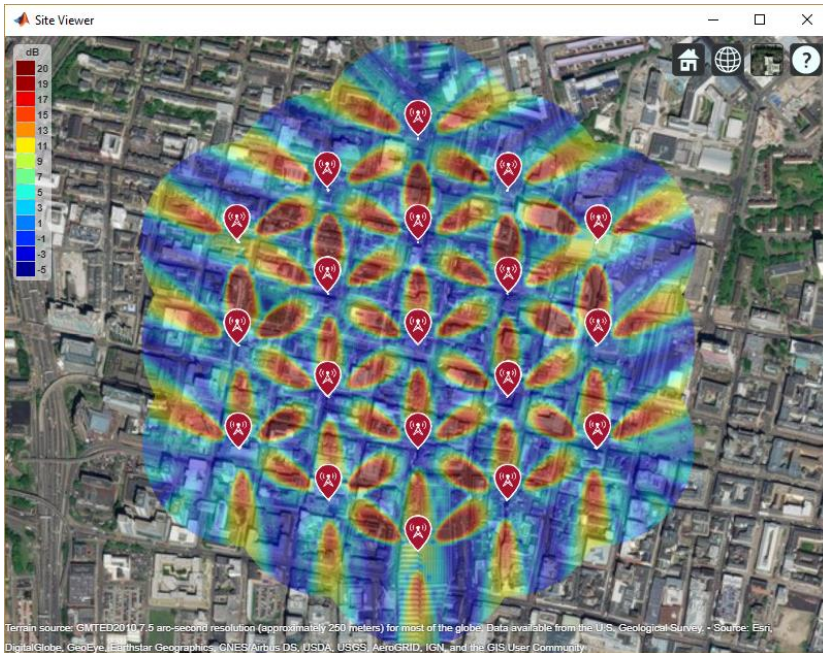


Hardware



What is the most basic way we can look at a wireless link?

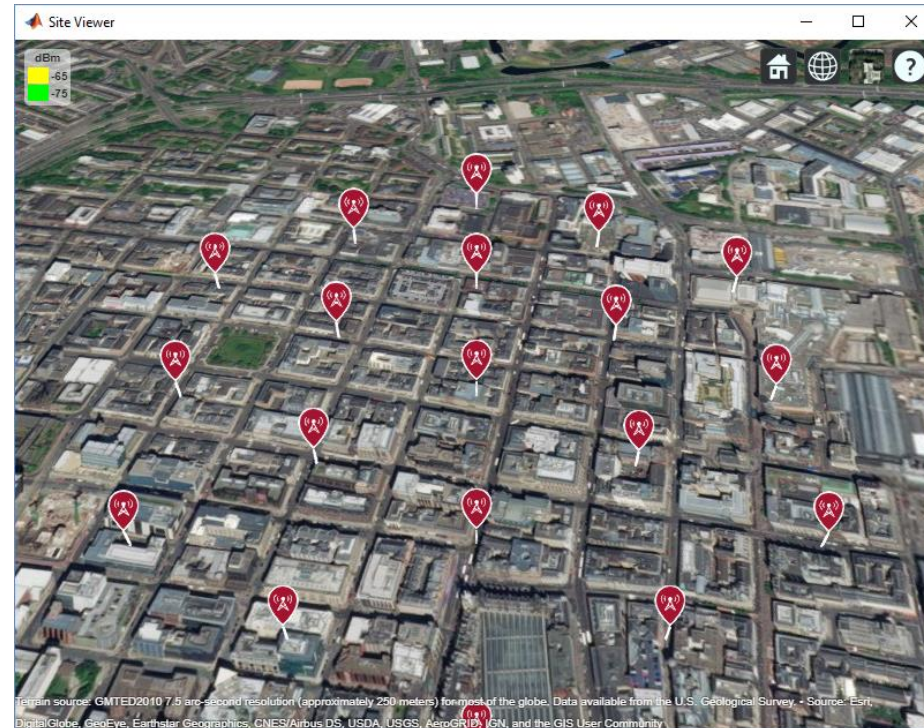
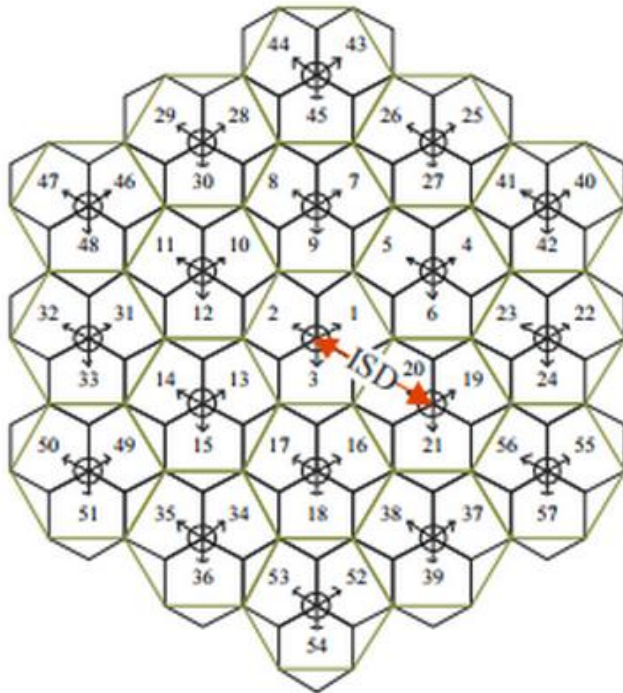
Scenario Modeling



- Scenario Level Modeling
 - RF propagation
 - Multi-transmitter scenarios
 - Coverage

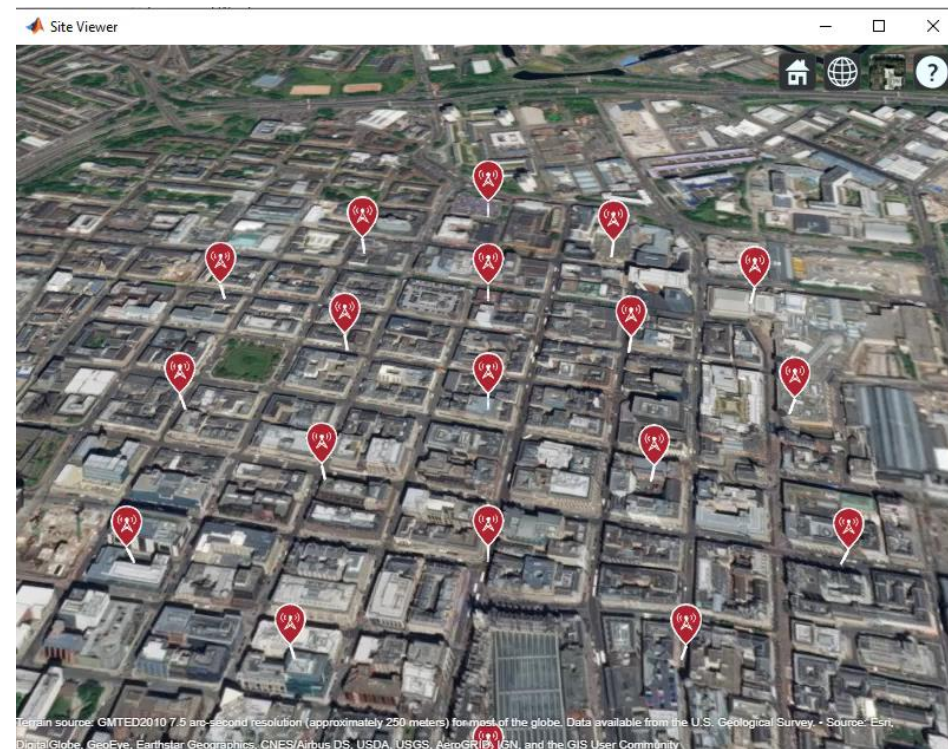
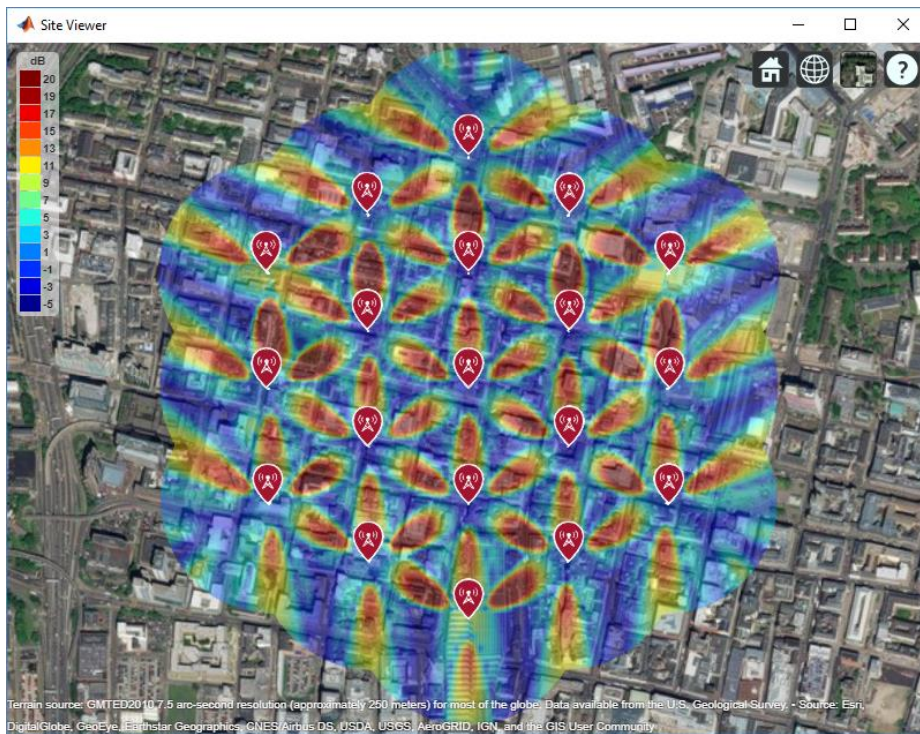
What relevant items need to be included to analyze a realistic 5G coverage scenario?

- Multiple Transmitter Scenario for analyzing SINR
 - Frequency = 4GHz
 - TX power = 44dBm
 - Antenna height = 25m
 - Model 19 adjacent cells
 - Each cell has 3 sectors



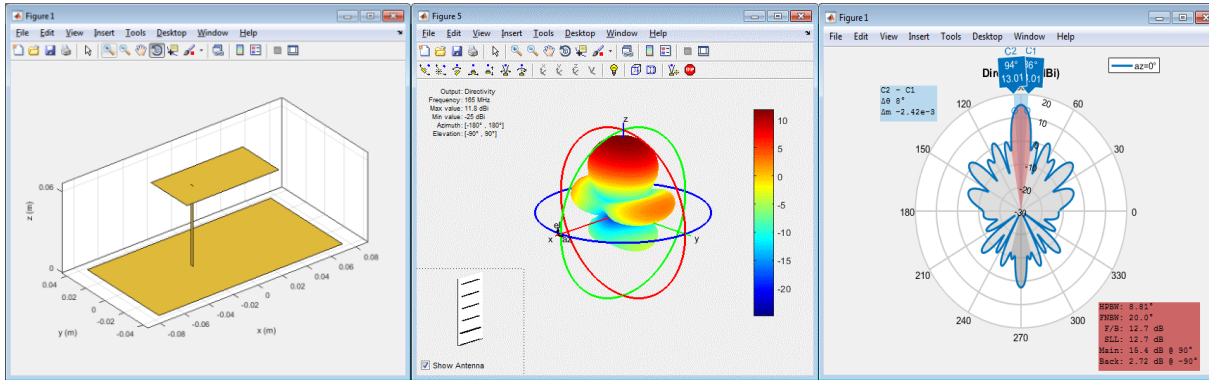
What are the different scenarios that can be analyzed?

- Select unique RF propagation scenarios such as 'Close-in' and 'Rain' propagation models.
- Choose different antenna elements and array configurations to maximize coverage.

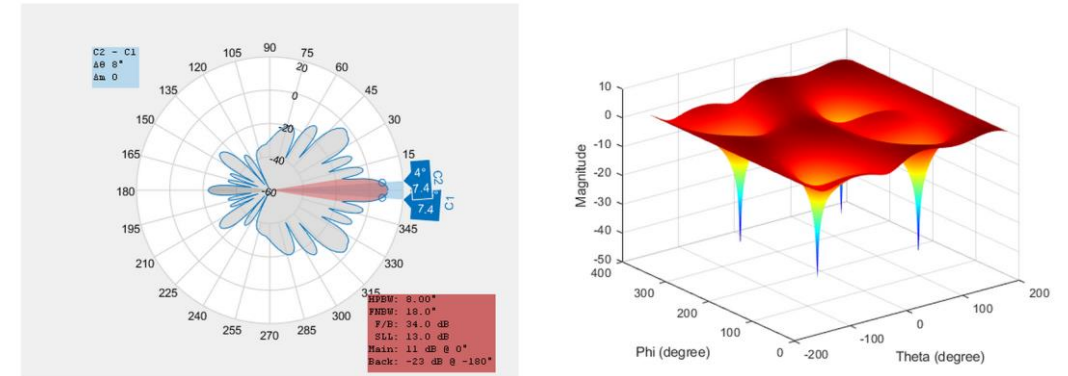


What are the different use cases for Antenna Toolbox?

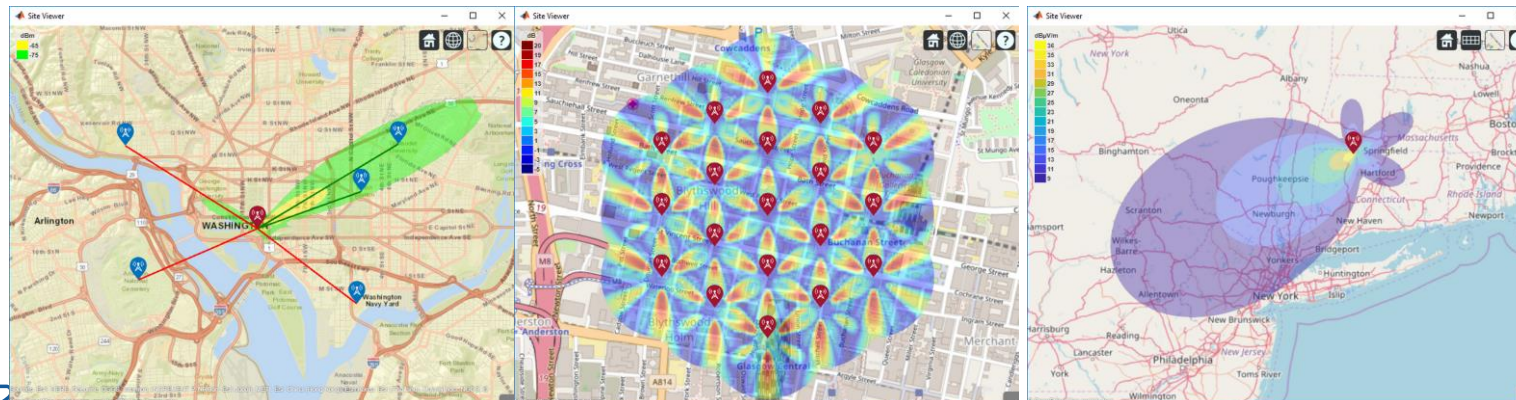
Antenna Element and Array Design



Visualization and Analysis of 3rd party Antenna Data



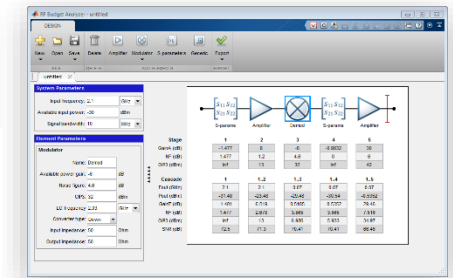
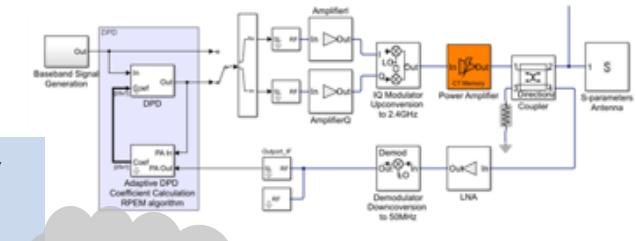
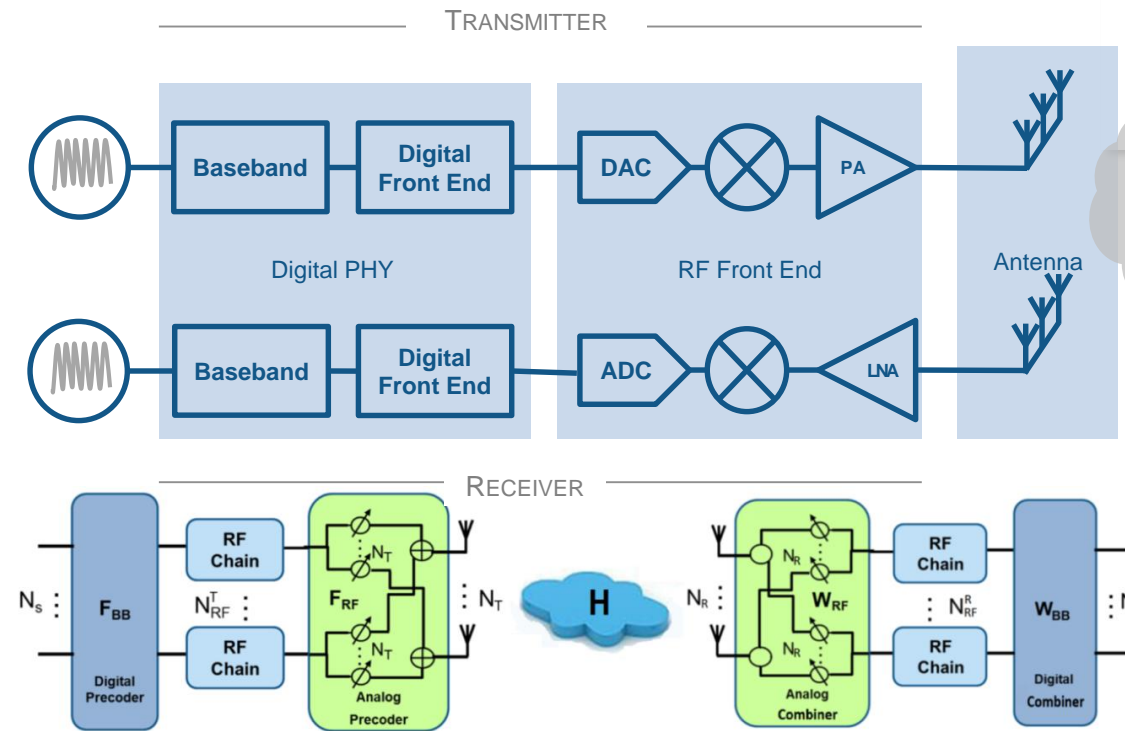
RF Propagation Visualization and Analysis



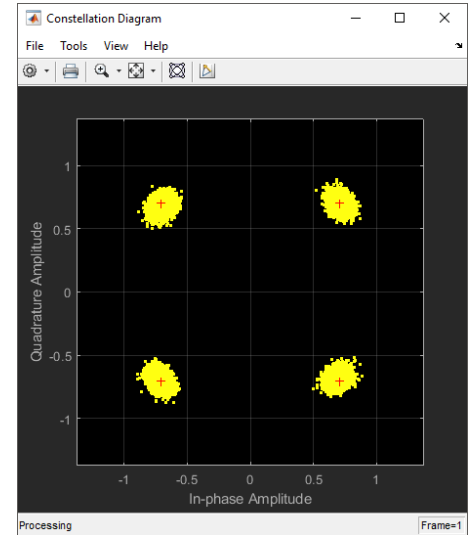
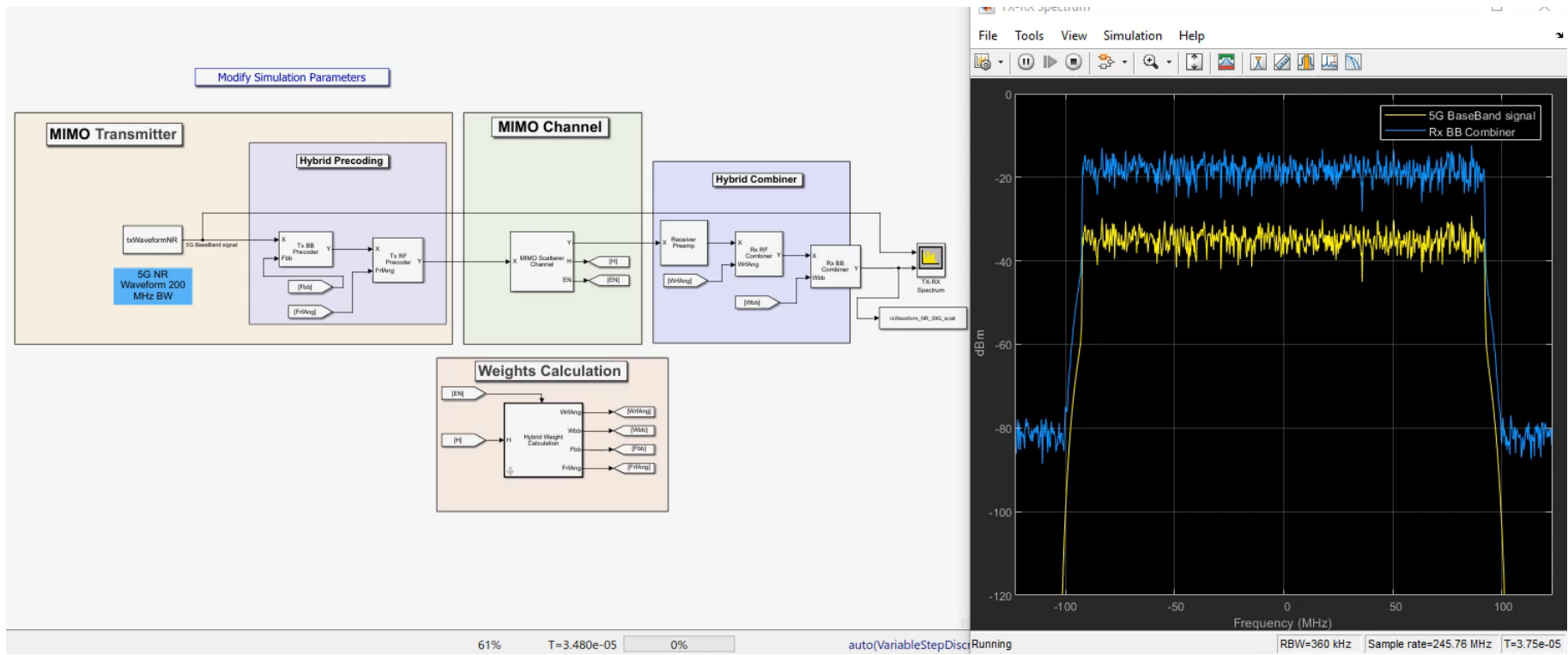
What type of fidelity do we want to add to a physical layer model?

- RF Front End
 - Noise budget
 - Gain
 - Non-linearity
 - Tx linearization
- Antennas
 - Arrays
 - Beamforming
 - Propagation effects
 - Loading

Link Level Modeling

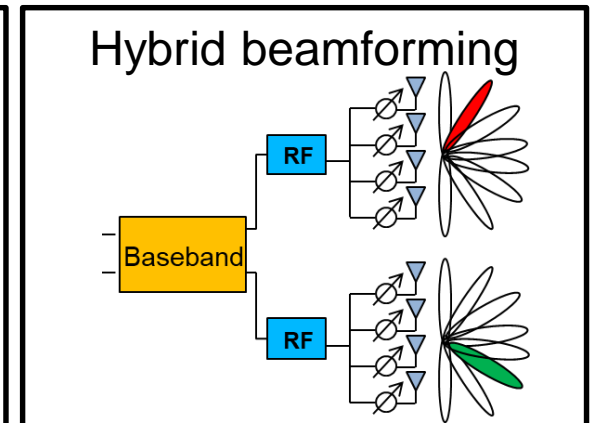
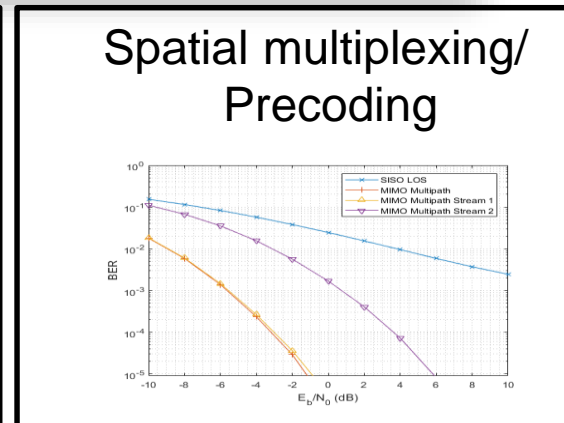
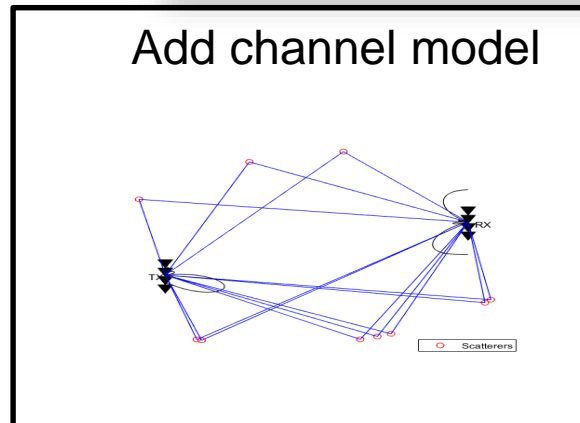
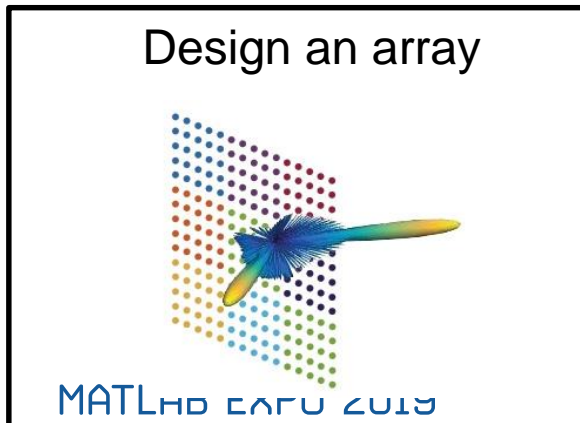
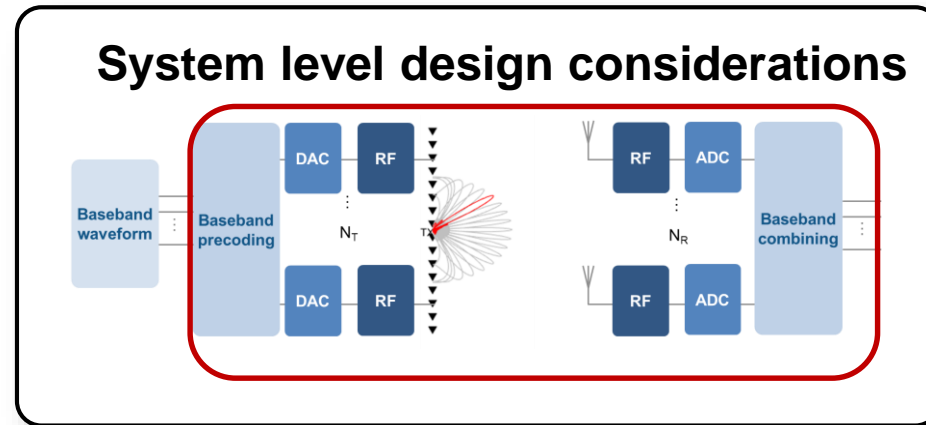


Why do link level modeling for a 5G mmWave system?



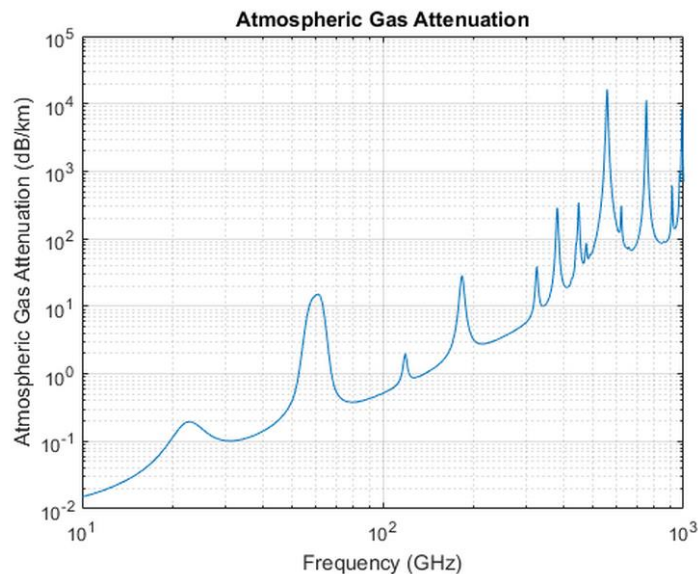
What needs to be included in a 5G system model to describe typical operation?

- Include fidelity that comprises of array behavior, channel modeling, spatial multiplexing and pre-coding and basic hybrid beamforming

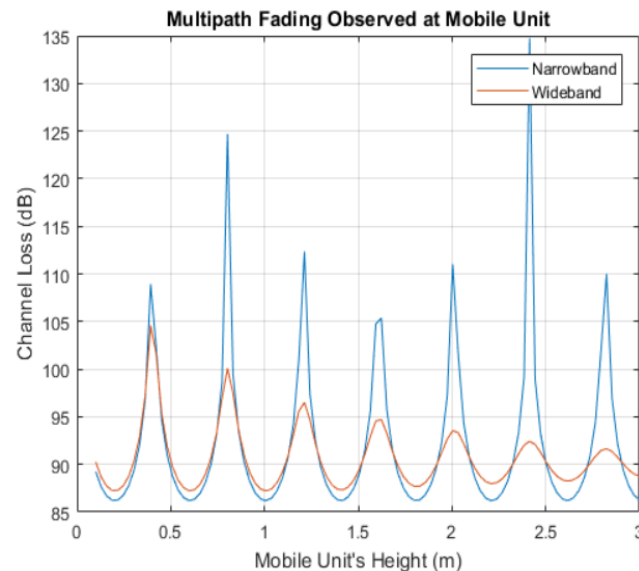


What comprises the behavior between the Tx and Rx antenna?

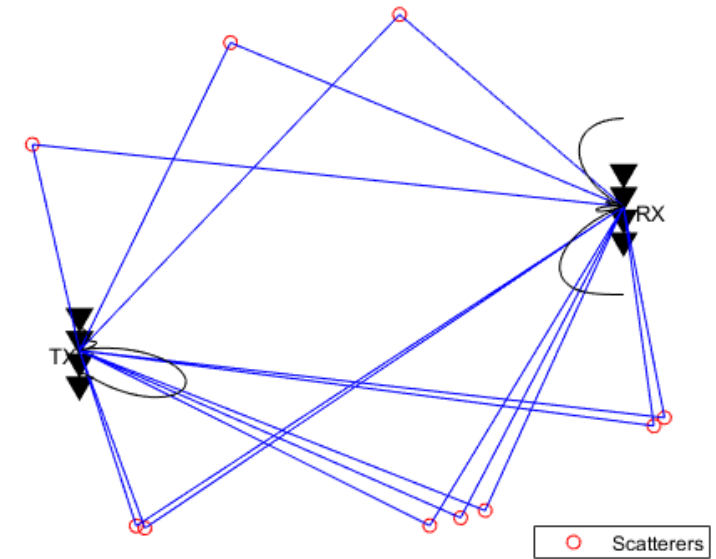
- Channel and RF propagation behavior



Signal Attenuation

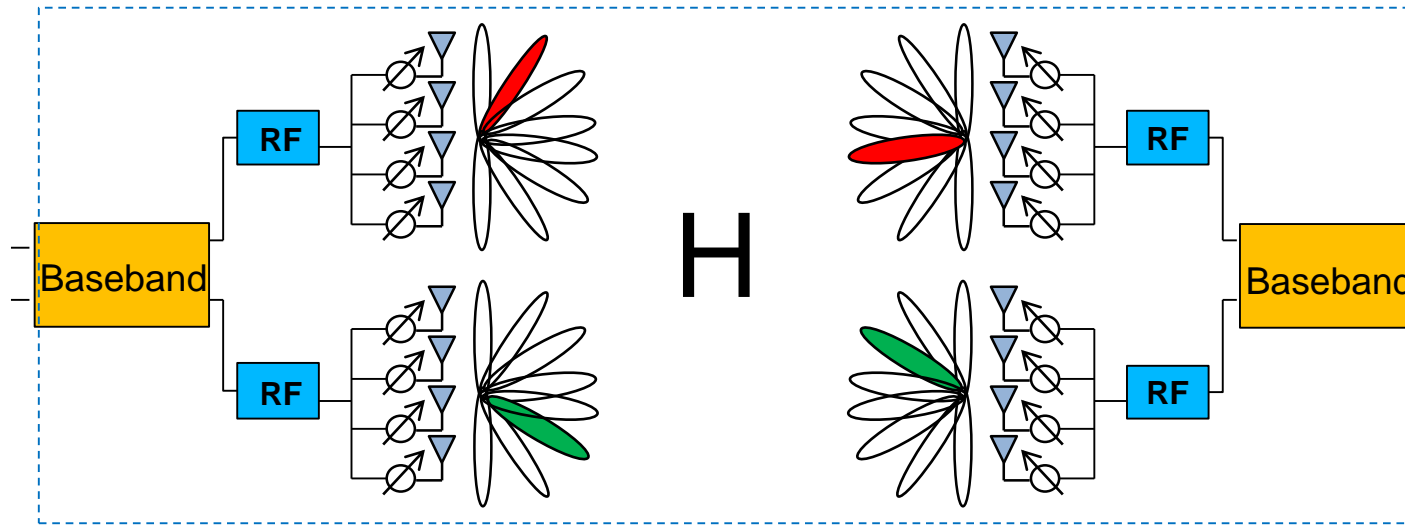


Wideband performance



Scatter-rich propagation

What is Hybrid Beamforming?

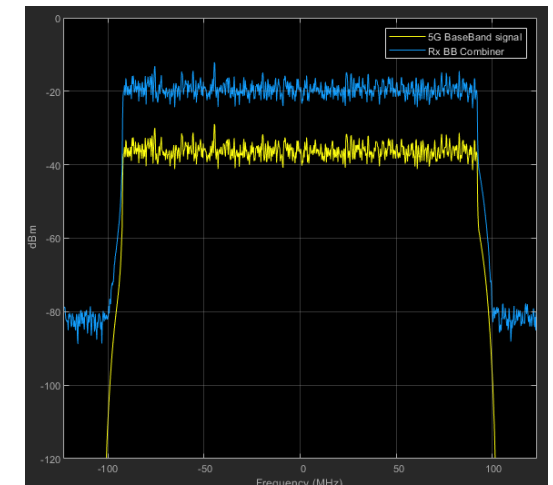
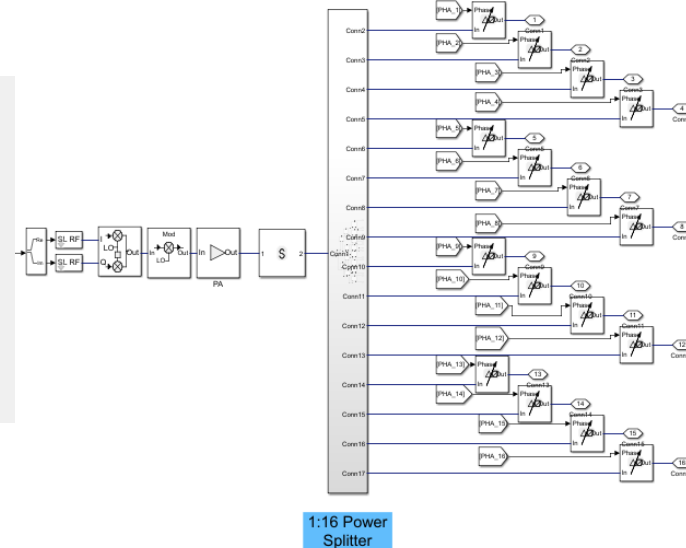
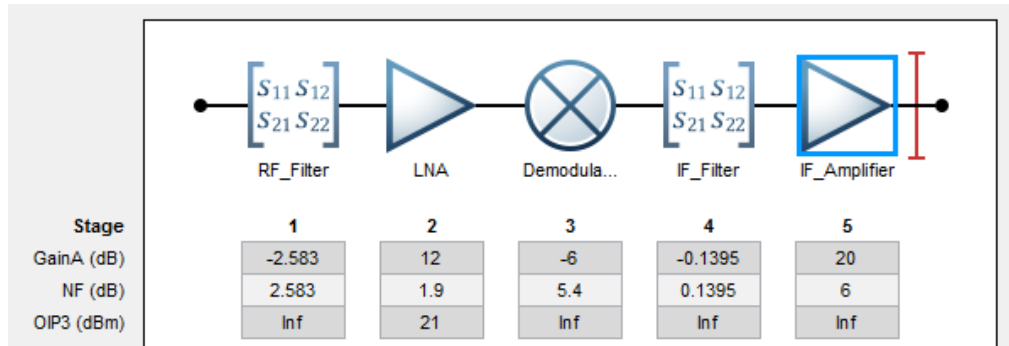


Beamforming done in two stages:

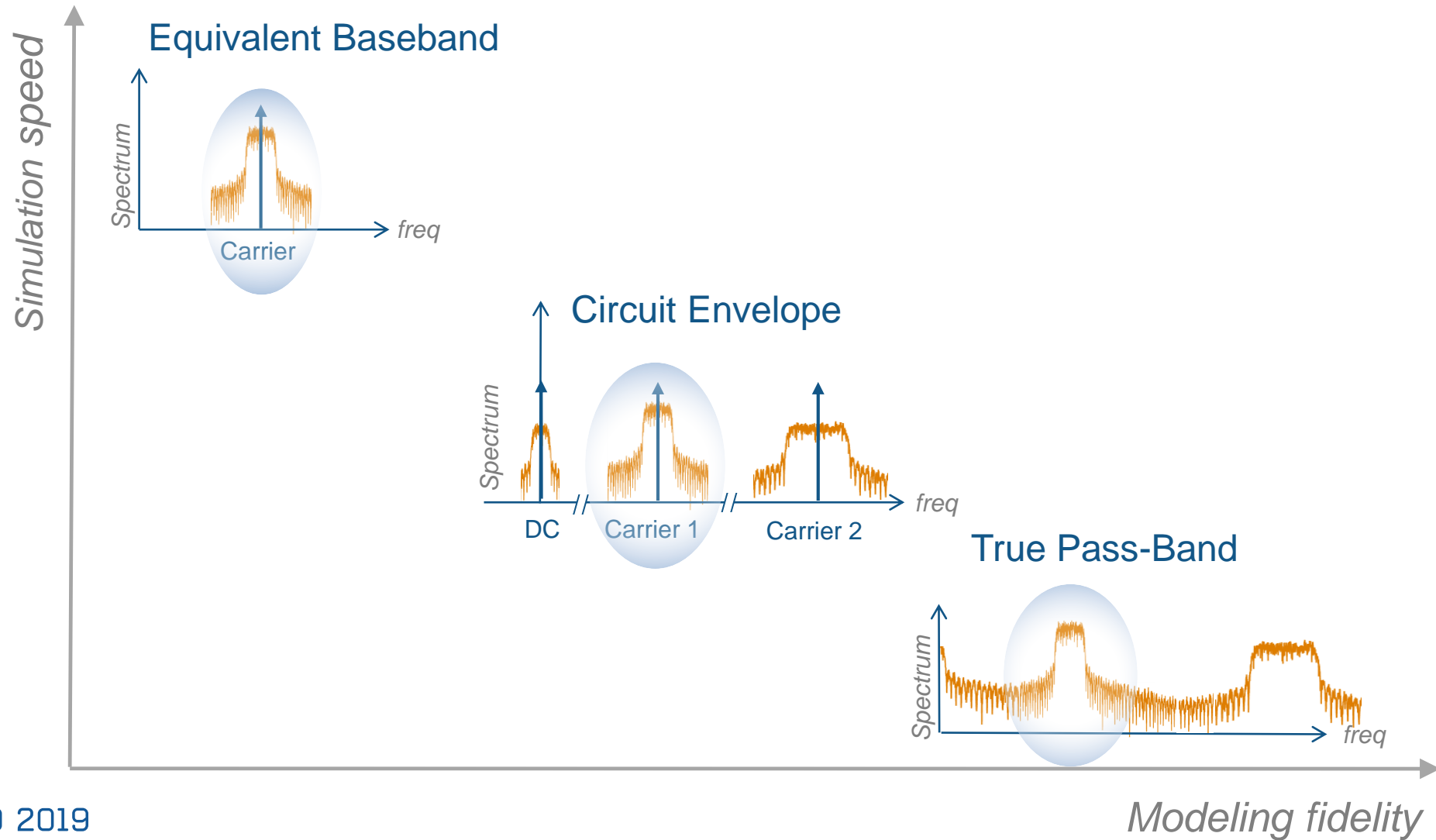
- RF Beamforming (phase shifters in RF front ends)
- Digital Beamforming (digital filtering of baseband signal)

Why do you want to add RF (System-Level) models to your PHY layer model?

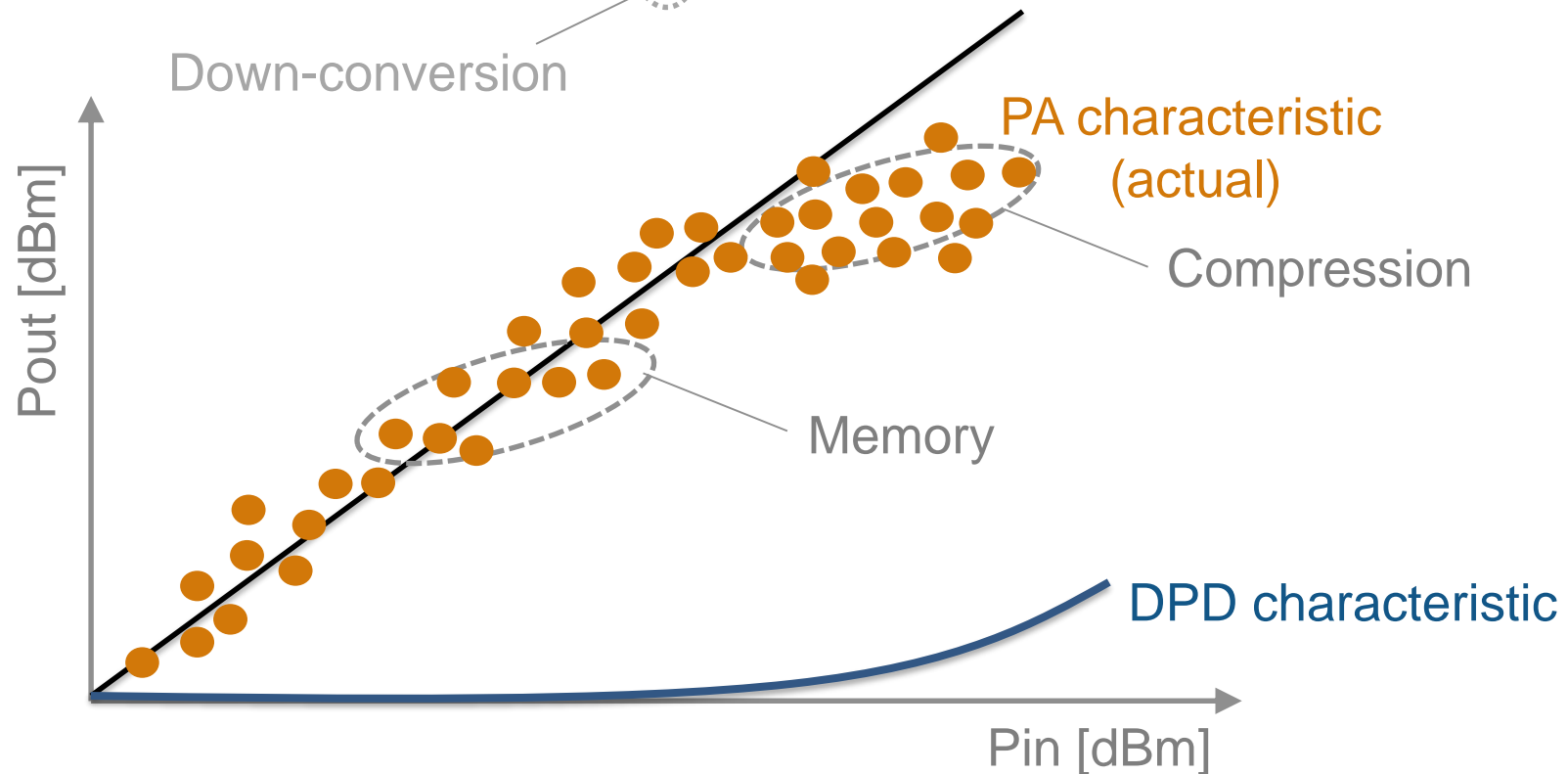
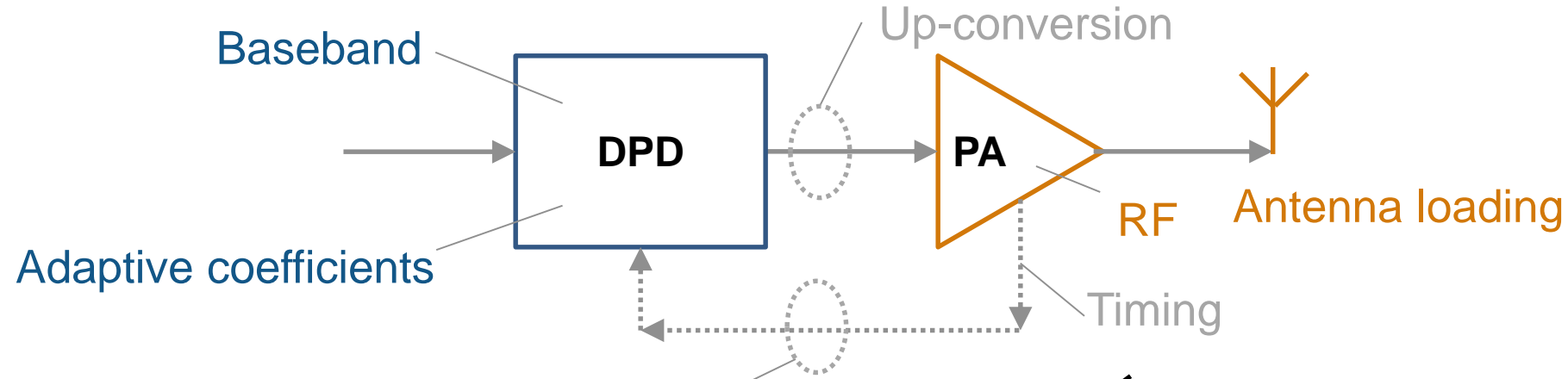
- Design the architecture and define the specs of the RF components
- Integrate RF front ends with adaptive algorithms such as DPD, AGC, beamforming
- Test and debug the implementation of the transceiver before going in the lab
- Use models and measured data to gain insights in your design
- Provide a model of the RF transceiver to your colleagues and customers



Circuit Envelope to Trade-off Fidelity and Speed



PA Linearization: Digital Pre Distortion (DPD) in Practice



PA Modeling Workflow

- Get I/Q (time domain, wideband) measurement data from your PA
- Fit the data with a memory polynomial (extract the coefficients) using MATLAB
- Verify the quality of the polynomial fitting (time, frequency)

$$y_{\text{MP}}(n) = \sum_{k=0}^{K-1} \sum_{m=0}^{M-1} a_{km} x(n-m) |x(n-m)|^k .$$

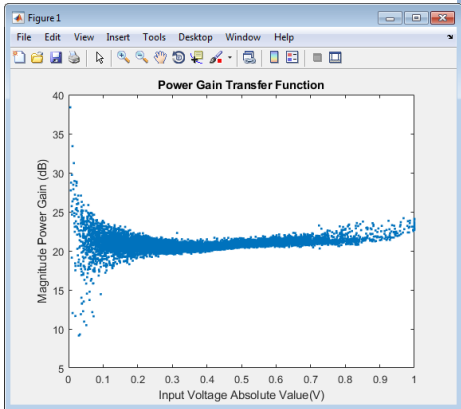
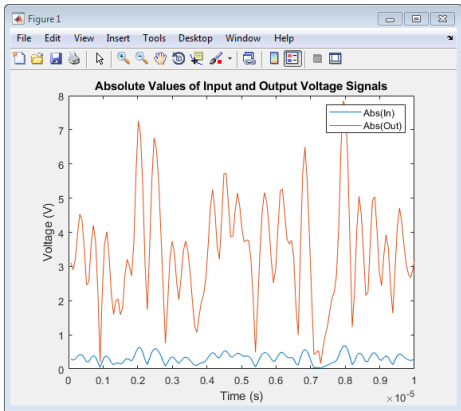
Memory length →

Order →

9.4522 + 24.3710i	8.3372 + 22.5027i	-7.6555 - 17.8049i	5.2338 + 12.8109i	-3.5523 - 8.3659i	1.4949 + 4.0988i	-0.6511 - 1.0900i
15.8350 + 25.6405i	3.8876 + 1.8345i	-3.1046 + 0.5440i	2.1230 + 0.9708i	1.0384 - 2.0353i	2.5988 + 0.4408i	1.6011 - 0.5171i
-67.4772 - 80.6146i	-20.3301 - 13.0211i	13.5985 + 0.1138i	-6.0557 - 2.5104i	-2.4325 + 4.5629i	-7.4792 - 0.7205i	-4.3852 - 0.3074i

What resources are available to characterize a PA Model?

PA Data



MATLAB fitting procedure (White box)

```
function a_coef = fit_memory_poly_model(x,y,memLen,degLen,modType)
% FIT_MEMORY_POLY_MODEL
% Procedure to compute a coefficient matrix given input and output
% signals, memory length, nonlinearity degree, and model type.
%
% Copyright 2017 MathWorks, Inc.

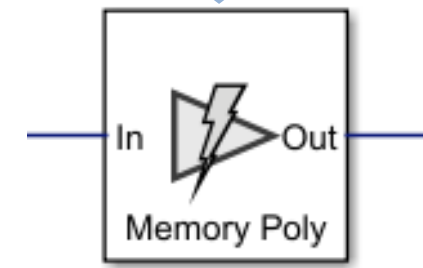
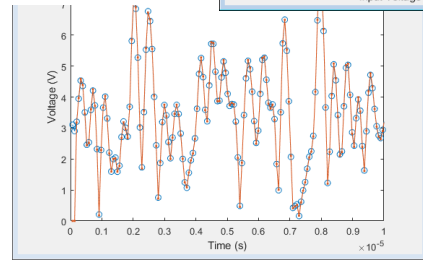
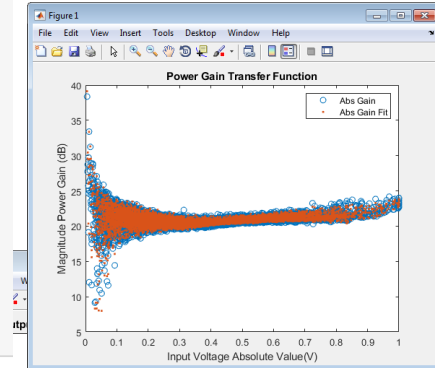
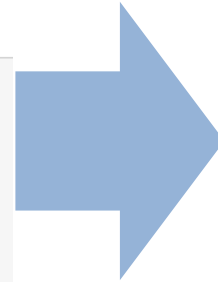
x = x(:);
y = y(:);
xLen = length(x);

switch modType
case 'memPoly' % Memory polynomial
xrow = reshape((memLen:-1:1)' + (0:xLen:xLen*(degLen-1)),1,[]);
xVec = (0:xLen-memLen)' + xrow;
xPow = x.*(abs(x).^(0:degLen-1));
xVec = xPow(xVec);
case 'ctMemPoly' % Cross-term memory polynomial
absPow = (abs(x).^(1:degLen-1));
partTop1 = reshape((memLen:-1:1)' + (0:xLen:xLen*(degLen-2)),1,[]);
topPlane = reshape(
[ones(xLen-memLen+1,1),absPow((0:xLen-memLen)' + partTop1)].', ...
1,memLen*(degLen-1)+1,xLen-memLen+1);
sidePlane = reshape(x((0:xLen-memLen)' + (memLen:-1:1)).', ...
memLen,1,xLen-memLen+1);
cube = sidePlane.*topPlane;
xVec = reshape(cube,memLen*(memLen*(degLen-1)+1),xLen-memLen+1).';
end

coef = xVec\y(memLen:xLen);
a_coef = reshape(coef,memLen,numel(coef)/memLen);
```

PA model coefficients

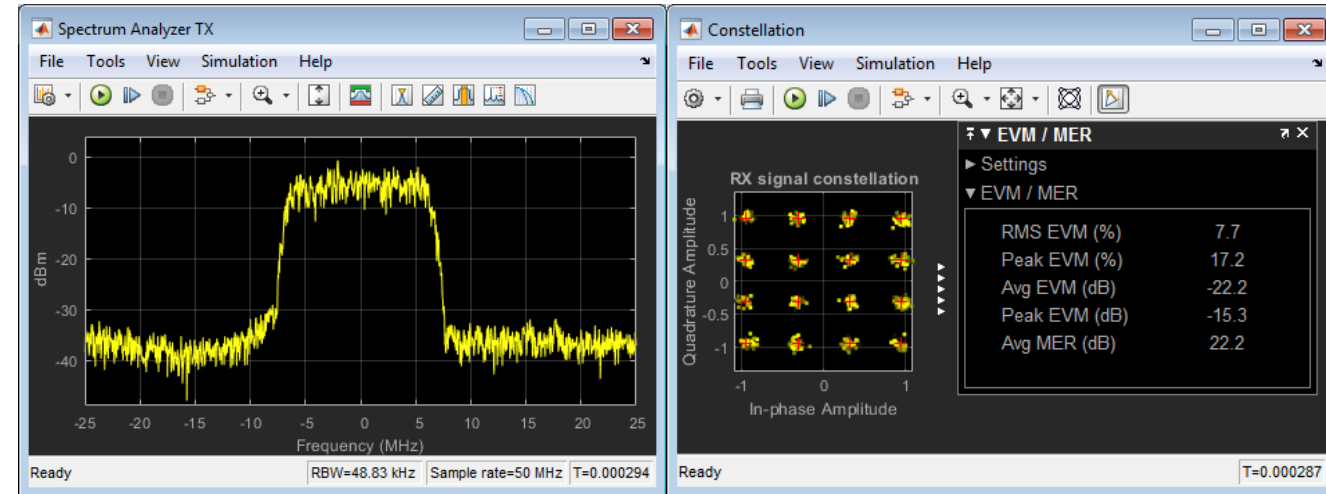
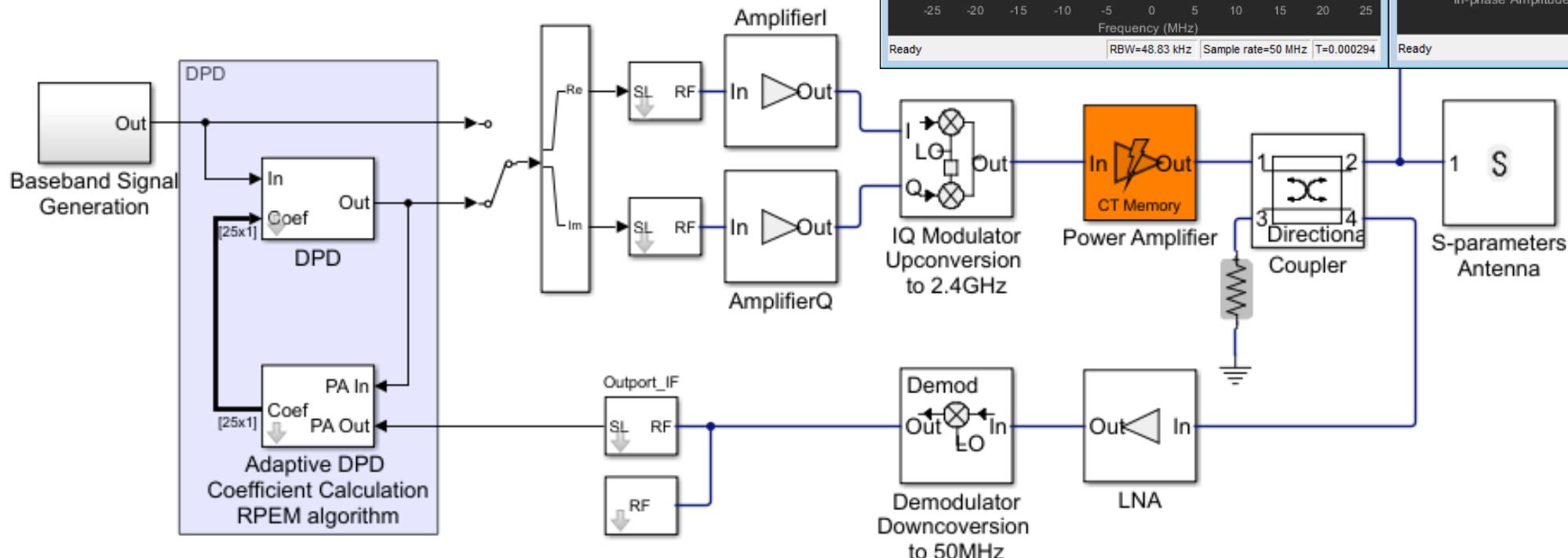
	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	7.1756 + 1.1238i	57.1783 - 12.3324i	10.5876 - 7.5994i	-2.423... - 4.379... - 1.125... 24.61... 1.461... 4.390... -94.35... -2.338... -8.825... 1.934... 1.8...										
2	3.2336 - 0.7538i	-25.2834 + 7.1506i	-4.4593 + 13.8723i	-9.675... 2.191... 2.847... 1.131... -8.420... -9.565... -4.801... 1.563... 2.309... 9.079... -1.4...										
3	-1.6834 + 1.1150i	12.5544 - 6.4201i	-4.6721 - 4.7128i	16.98... -1.006... 51.69... -1.516... 3.683... -2.068... 5.637... -6.580... 3.495... -9.910... 5.7...										
4														
5														
6														
7														
8														



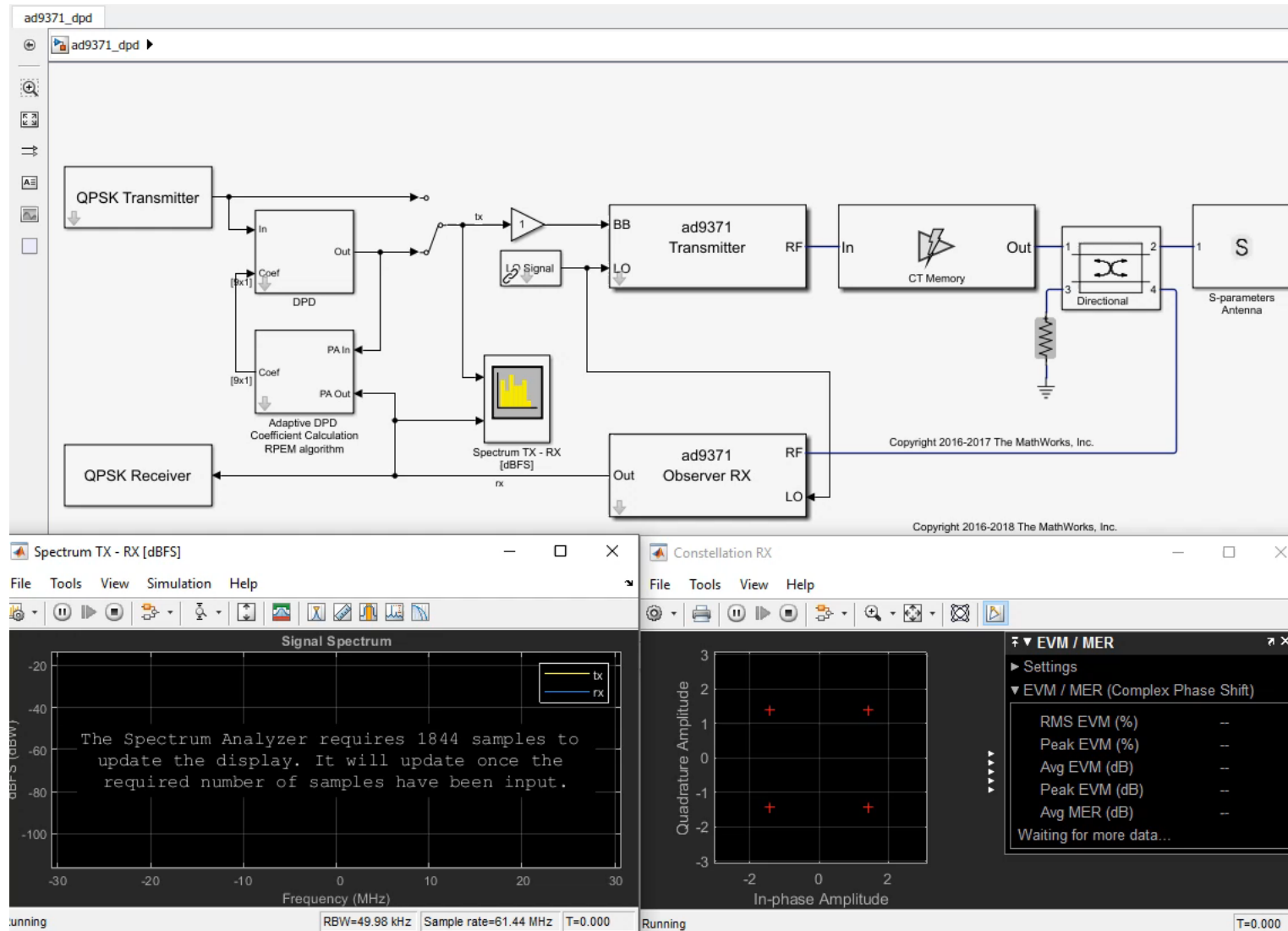
PA model for circuit envelope simulation

Why is static DPD modeling not enough for 5G systems?

- Circuit Envelope for fast RF simulation
- Low-power RF and analog components
 - Up-conversion / down-conversion
 - Antenna load
- Digital signal processing algorithm: DPD



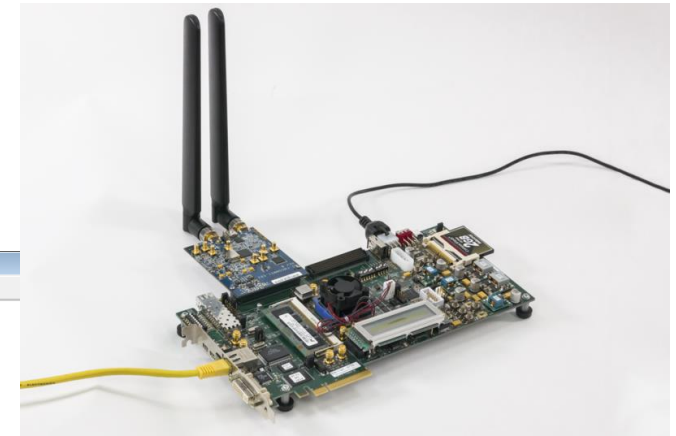
Real-Life Example: AD9371 Transmitter + Observer



From Simulation to Implementation: HDL Code Generation

Automatically generate synthesizable HDL (Verilog / VHDL) code

- Make your model hardware “friendly”
- Estimate utilized resources
- Optimize model and generated code (speed, cost)
- Target FPGAs for rapid prototyping



Generic Resource Report for PA_DPD_ImplementationFixed_17b	
Summary	
Multipliers	240
Adders/Subtractors	348
Registers	8
Total 1-Bit Registers	128
	0
	250
	868
	0
	0

```

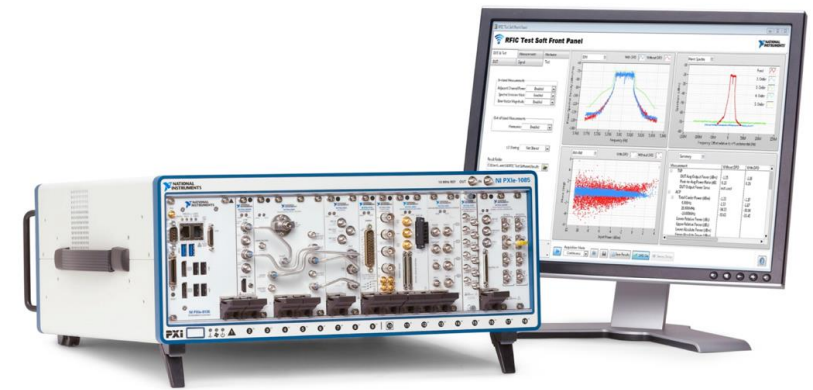
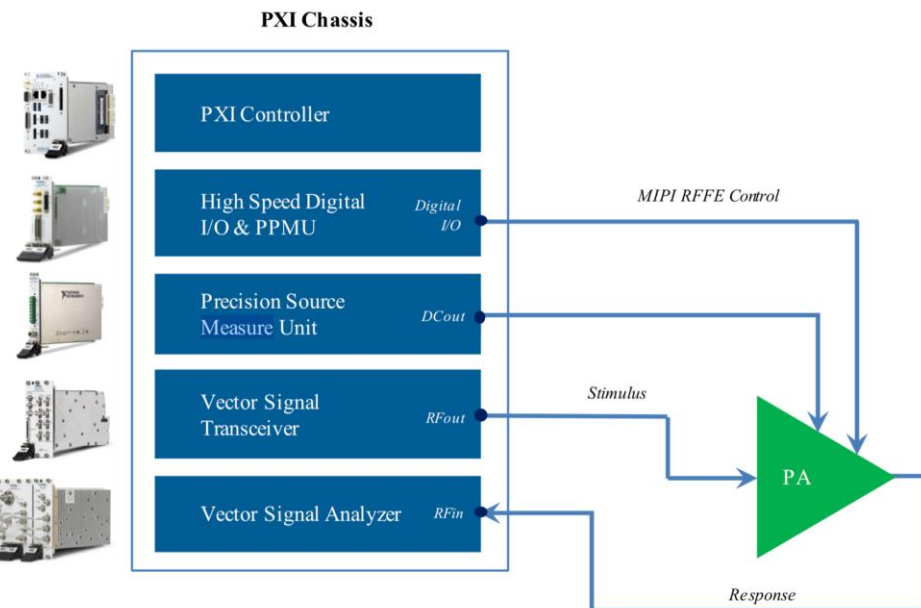
124
125 assign v1_re_1 = v1_re[1];
126
127 assign v1_re_2 = v1_re[2];
128
129 assign v1_re_3 = v1_re[3];
130
131 assign v1_re_4 = v1_re[4];
132
133 assign Complex_to_Real_Imag_out2[0] = Inl_im_0;
134 assign Complex_to_Real_Imag_out2[1] = Inl_im_1;
135 assign Complex_to_Real_Imag_out2[2] = Inl_im_2;
136 assign Complex_to_Real_Imag_out2[3] = Inl_im_3;
137 assign Complex_to_Real_Imag_out2[4] = Inl_im_4;
138
139 // cs43/Abs1
140 assign Abs1_cast = (Complex_to_Real_Imag_out2[0][15], Complex_to_Real_Imag_out2[0]);
141 assign Abs1_y[0] = (Complex_to_Real_Imag_out2[0] < 16'sb0000000000000000 ? - (Abs1_cast) :
142 {Complex_to_Real_Imag_out2[0][15], Complex_to_Real_Imag_out2[0]});
143 assign Abs1_cast_1 = (Complex_to_Real_Imag_out2[1][15], Complex_to_Real_Imag_out2[1]);
144 assign Abs1_y[1] = (Complex_to_Real_Imag_out2[1] < 16'sb0000000000000000 ? - (Abs1_cast_1) :
145 {Complex_to_Real_Imag_out2[1][15], Complex_to_Real_Imag_out2[1]});
146 assign Abs1_cast_2 = (Complex_to_Real_Imag_out2[2][15], Complex_to_Real_Imag_out2[2]);
147 assign Abs1_y[2] = (Complex_to_Real_Imag_out2[2] < 16'sb0000000000000000 ? - (Abs1_cast_2) :
148 {Complex_to_Real_Imag_out2[2][15], Complex_to_Real_Imag_out2[2]});
149 assign Abs1_cast_3 = (Complex_to_Real_Imag_out2[3][15], Complex_to_Real_Imag_out2[3]);
150 assign Abs1_y[3] = (Complex_to_Real_Imag_out2[3] < 16'sb0000000000000000 ? - (Abs1_cast_3) :
151 {Complex_to_Real_Imag_out2[3][15], Complex_to_Real_Imag_out2[3]});
152 assign Abs1_cast_4 = (Complex_to_Real_Imag_out2[4][15], Complex_to_Real_Imag_out2[4]);
153 assign Abs1_y[4] = (Complex_to_Real_Imag_out2[4] < 16'sb0000000000000000 ? - (Abs1_cast_4) :
154 {Complex_to_Real_Imag_out2[4][15], Complex_to_Real_Imag_out2[4]});
155 assign Abs1_out1[0] = (Abs1_y[0][14:0], 1'b0);
156 assign Abs1_out1[1] = (Abs1_y[1][14:0], 1'b0);
157 assign Abs1_out1[2] = (Abs1_y[2][14:0], 1'b0);
158 assign Abs1_out1[3] = (Abs1_y[3][14:0], 1'b0);
159 assign Abs1_out1[4] = (Abs1_y[4][14:0], 1'b0);
160
161
162 assign v1_im[0] = Abs1_out1[0];
163 assign v1_im[1] = Abs1_out1[1];
164 assign v1_im[2] = Abs1_out1[2];
165 assign v1_im[3] = Abs1_out1[3];
166 assign v1_im[4] = Abs1_out1[4];
167
168
169 assign v1_im_0 = v1_im[0];
170
171
    
```

How do we transition from software models to hardware?

- Implementing DPD in hardware
 - Data streaming
 - Prototype on hardware

```

1 function predistorted = dpd(xx, zz, y)
2 g = rms(y) / rms(zz);
3 yy = y / g;
4 Y = [yy, yy.*abs(yy), yy.*(abs(yy)).^2, yy.*(abs(yy)).^3, yy.*(abs
5 a = Y \ zz;
6 predistorted = [xx, xx.*abs(xx), xx.*(abs(xx)).^2, xx.*(abs(xx)).^
7
    
```



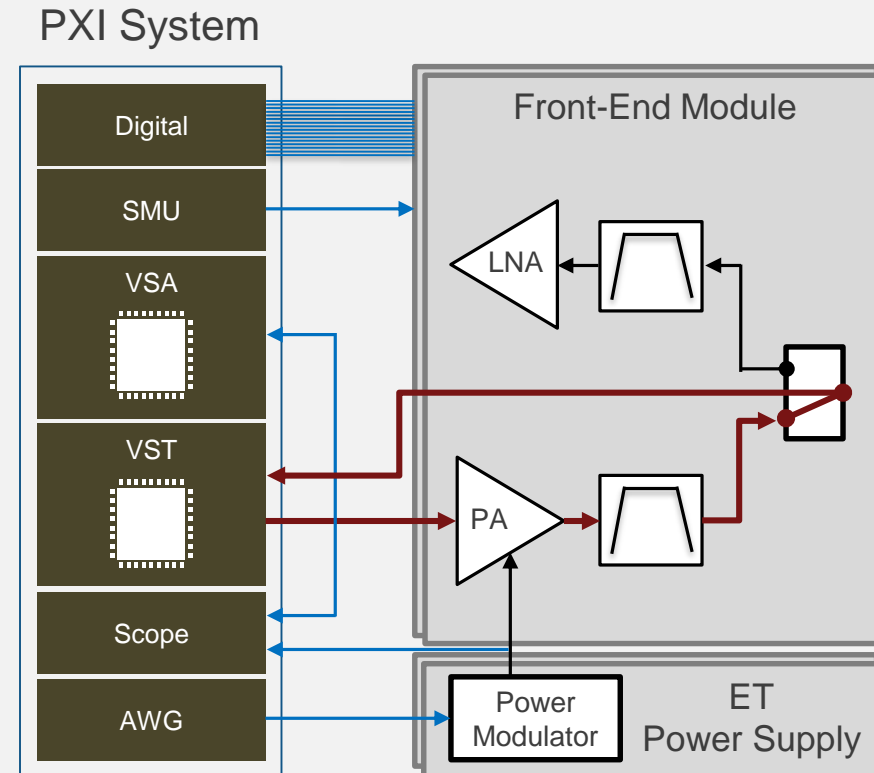
Hardware

Connecting System-Level Models to Hardware for Design and Verification

NI Front-End Module Test With DPD

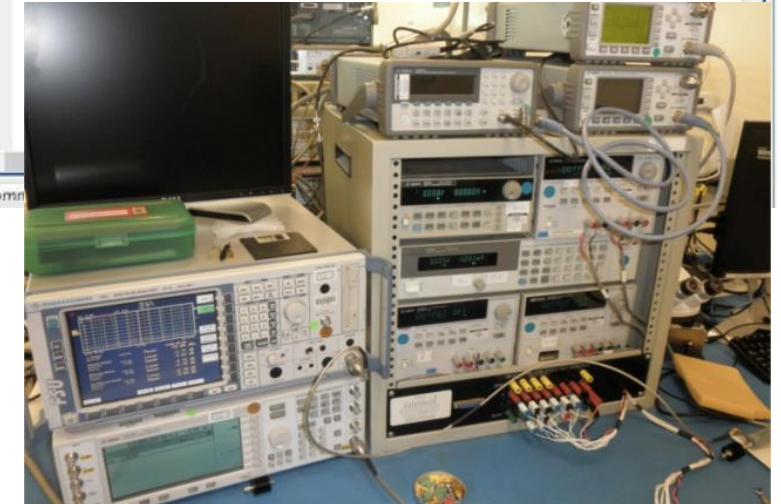
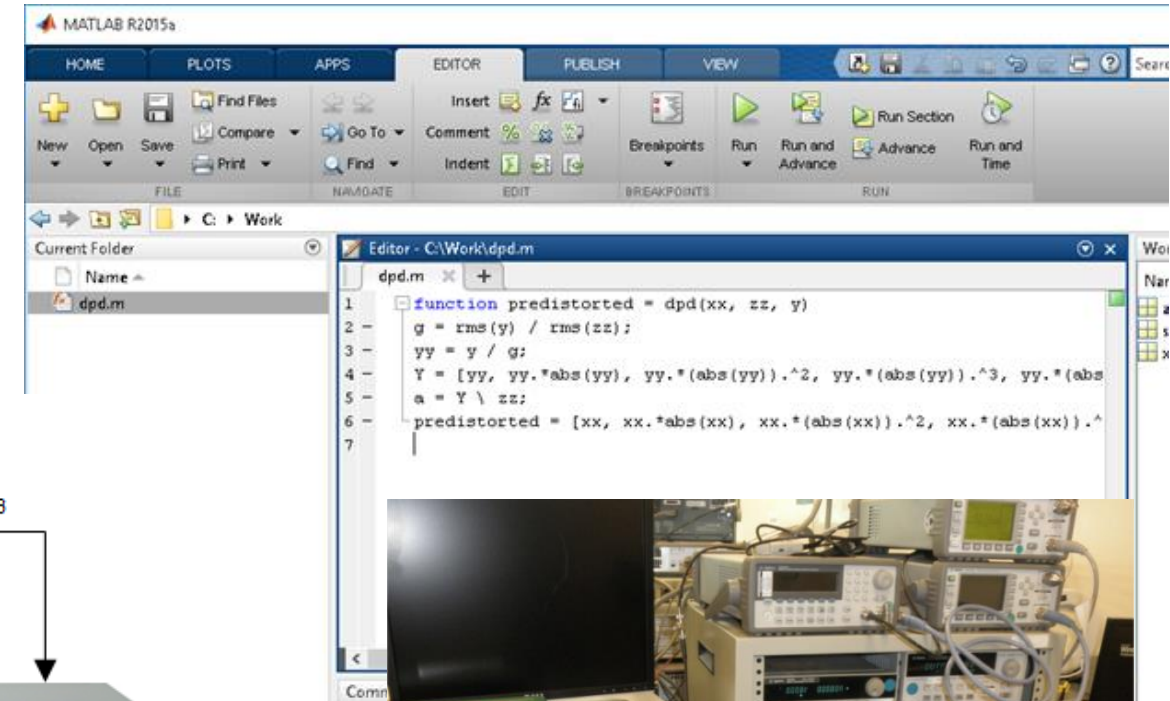
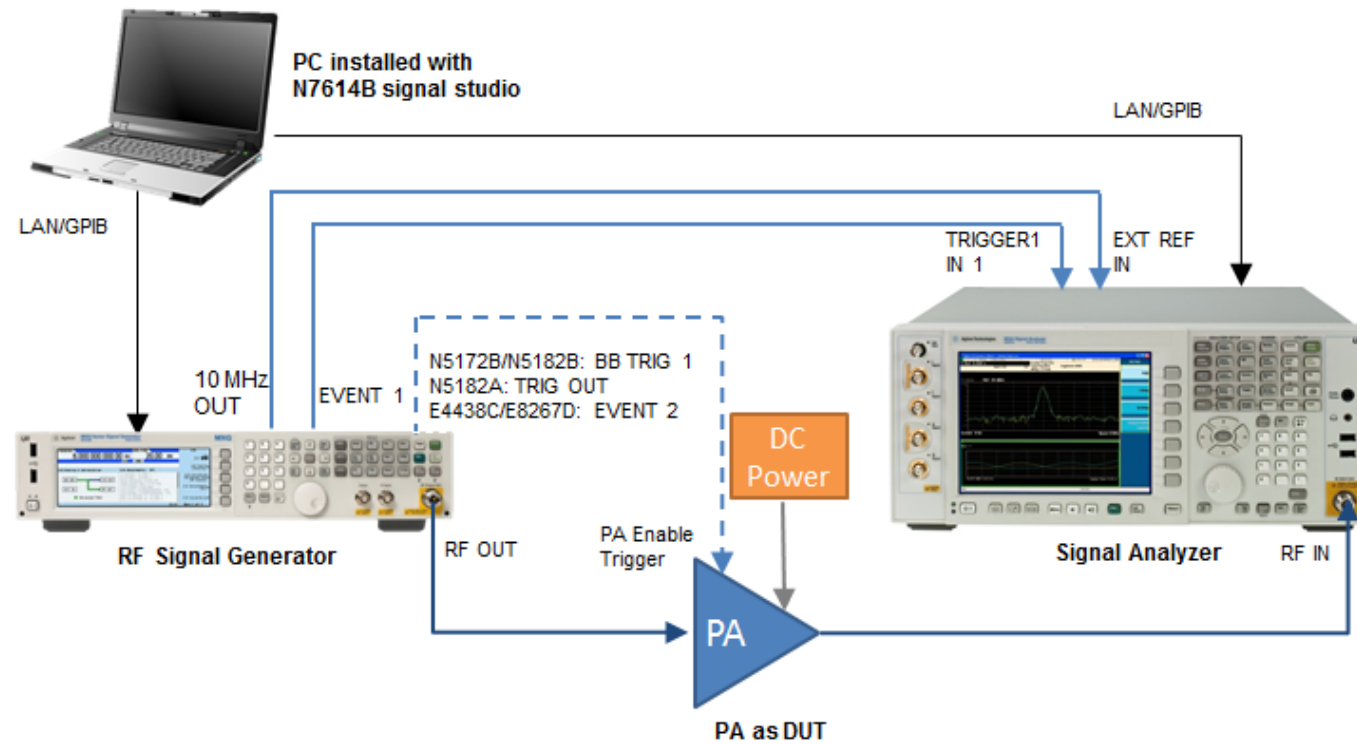
- VST with 1 GHz instantaneous generation and analysis bandwidth
- Free NI-RFmx SpecAn with LUT, MPM, and GMP DPD models
- Free RFIC Test Software with DPD automation examples

- 1 Generate reference waveform and acquire distorted waveform
- 2 Create predistortion model by comparing reference waveform to distorted waveform
- 3 Apply DPD to reference waveform using predistortion model
- 4 Generate predistorted waveform and make measurements



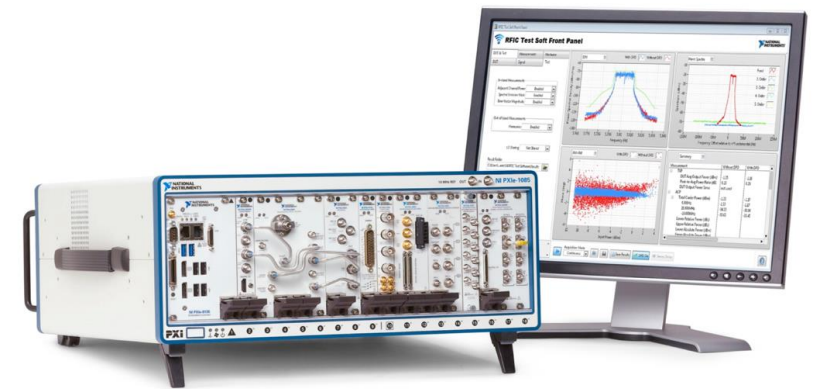
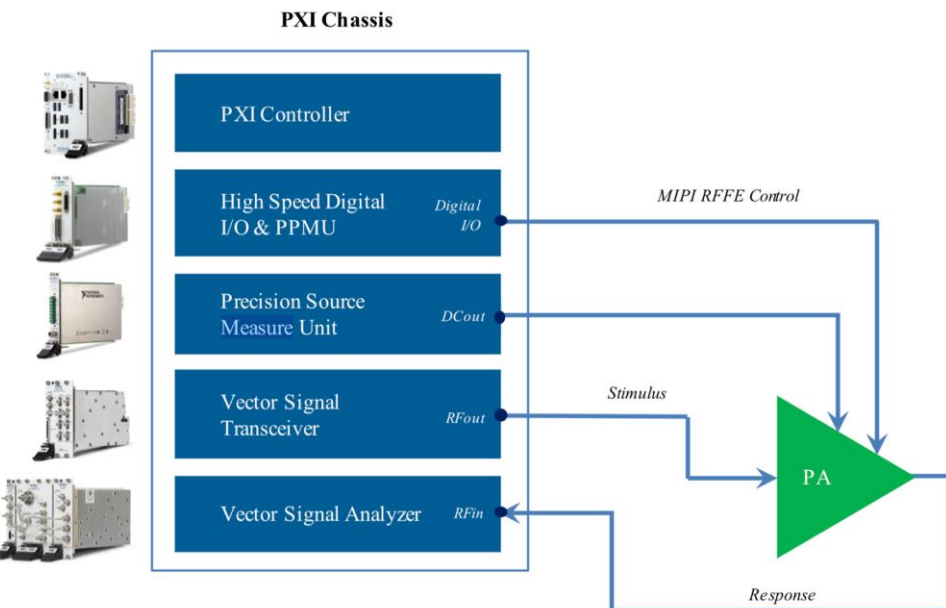
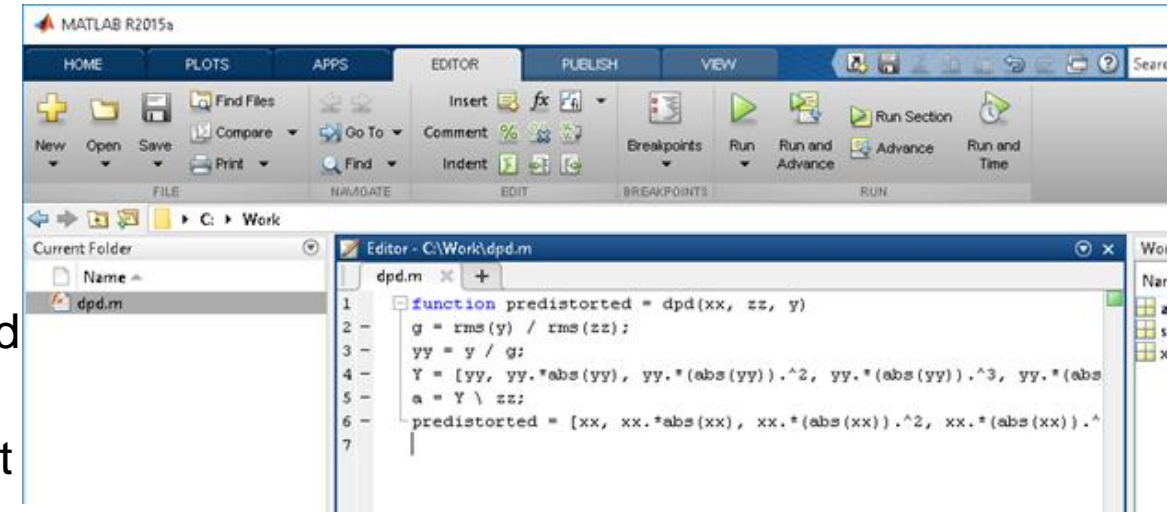
Traditional T&M Setup for MATLAB Based PA Characterization with DPD Algorithm Running in MATLAB

- Familiar user experience for many engineers
- Slower measurement speed, Large physical footprint
- Expensive to upgrade or replace – even Software
- Difficult to synchronize for ET & DPD
- Tradeoffs between speed and accuracy

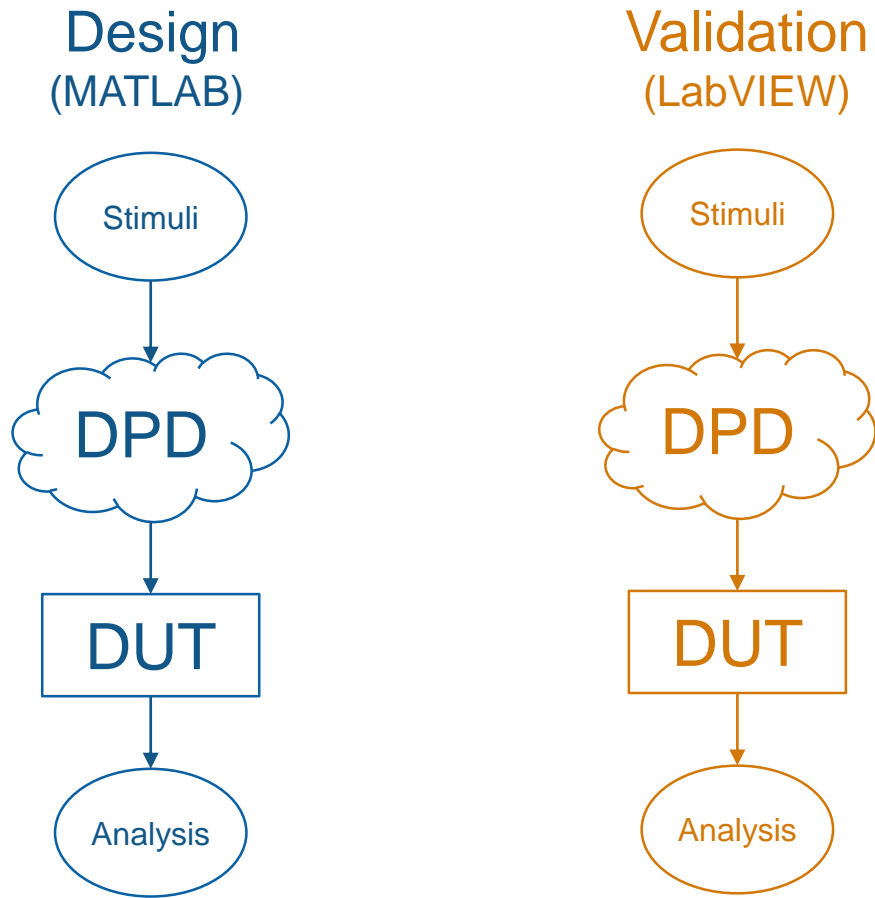


NI PXI Setup for MATLAB Based PA Characterization with DPD & ET Algorithm Running in MATLAB

- Similar user experience as box-instruments
- Faster and FPGA-accelerated measurement speed, at a fraction of the physical footprint
- Modularity for incremental upgrades
- Native synchronization technologies at sub nanosecond accuracy
- R&D grade measurement accuracy with production test speed

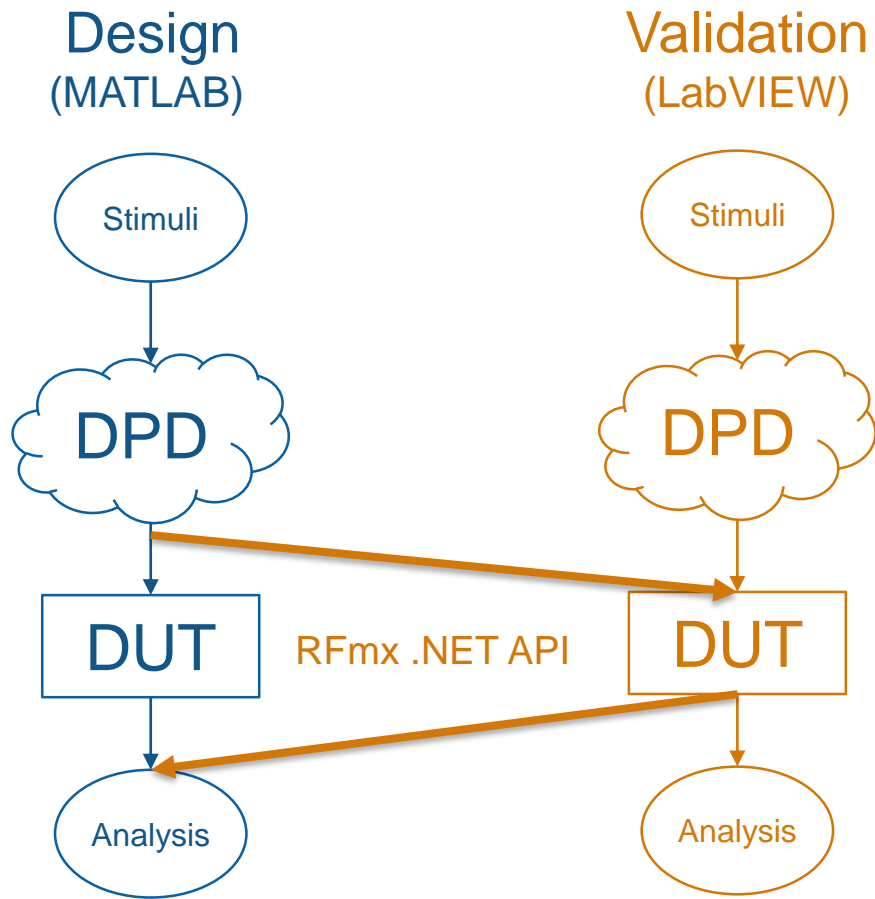


Enabling Integrated Semi PA Design & Validation Flow Between LabVIEW & MATLAB



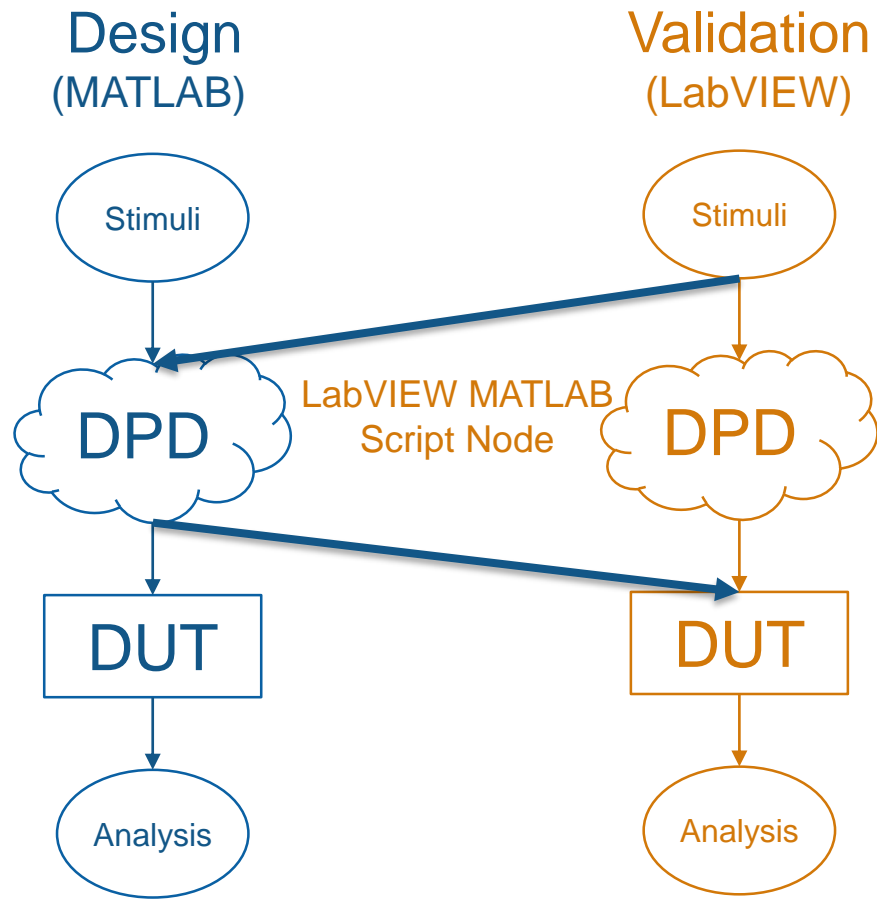
	Design (Sim-only)	V&V (T&M Only)
Waveform Generation	MATLAB	LabVIEW RFmx
DPD Algorithm	MATLAB (Custom)	RFmx + NanoSemi
DUT	Sim Model	Real
Waveform Analysis	MATLAB	LabVIEW RFmx
GUI environment	MATLAB	LabVIEW RFIC

Enabling Integrated Semi PA Design & Validation Flow Between LabVIEW & MATLAB



	Design (Sim-only)	V&V (T&M Only)	Design (Integrated)
Waveform Generation	MATLAB	LabVIEW RFmx	MATLAB
DPD Algorithm	MATLAB (Custom)	RFmx + NanoSemi	MATLAB (Custom)
DUT	Sim Model	Real	Real
Waveform Analysis	MATLAB	LabVIEW RFmx	MATLAB
GUI environment	MATLAB	LabVIEW RFIC	MATLAB

Enabling Integrated Semi PA Design & Validation Flow Between LabVIEW & MATLAB



	Design (Sim-only)	V&V (T&M Only)	Design (Integrated)	V&V (Integrated)
Waveform Generation	MATLAB	LabVIEW RFmx	MATLAB	LabVIEW RFmx
DPD Algorithm	MATLAB (Custom)	RFmx + NanoSemi	MATLAB (Custom)	MATLAB (Custom)
DUT	Sim Model	Real	Real	Real
Waveform Analysis	MATLAB	LabVIEW RFmx	MATLAB	LabVIEW RFmx
GUI environment	MATLAB	LabVIEW RFIC	MATLAB	LabVIEW RFIC

High-Power PA w/ DPD HW Demo Setup

PXIe-1078 Chassis
PXIe-8840 Controller
PXIe-5840 VST
PXIe-4112 Power Supply

Wideband LDMOS Two-stage Integrated Power Amplifier 20 W + 40 W, 28 V, 1805 – 2200 MHz

SKU: PTNC210604MD-V1

The PTNC210604MD is a wideband, two-stage, LDMOS integrated power amplifier. It incorporates internal matching for operation from 1805 to 2200 MHz, and dual independent outputs with 20 W and 40 W of output power each. It is available in a 14-lead plastic overmold package with gull wing leads.

Features

- On-chip matching for broadband operation
- Typical CW performance, 2200 MHz, 28 V, combined outputs
 - Output power at P3dB = 63 W
 - Linear Gain = 28 dB
 - Efficiency = 50.5%
- Capable of handling 10:1 VSWR @28 V, 10 W mod avg output power
- Integrated ESD protection
- Human Body Model Class 1A (per ANSI/ESDA/JEDEC JS-001)
- Integrated temperature compensation
- Pb-free and RoHS compliant



Peak Output Power	63W (P3dB)
Application	Telecom
Typical Power (PSAT)	20 + 40
Power Gain	27 dB
Operating Voltage	28 V
Frequency	1.8 - 2.2 GHz
Package Type	Surface Mount
Efficiency	37%
Technology	LDMOS



PA Design Engineer's View in MATLAB

The screenshot displays the MATLAB App Designer environment. On the left, the CODE BROWSER shows the following MATLAB code:

```

function [freq, spectrum] = FetchACP(app, selectorString, timeout)
    [-, spectrumNet] = app.nr.Acc.Results.FetchSpectrum(selectorString, timeout, []);
    spectrum = single(spectrumNet.GetData());
    freq = (0:single(spectrumNet.SampleCount) - 1) * spectrumNet.FrequencyIncrement + spectrumNet.StartFrequency;
end

function [i, q] = FetchConstellation(app, selectorString, timeout)
    import NationalInstruments.*;
    complexSingleArrayType = ComplexSingle.ComposeArray(8, 0);
    [-, constellation] = app.nr.ModAcc.Results.FetchPushDataConstellationTrace(selectorString, timeout, complexSingleArrayType);
    [i, q] = ComplexSingle.DecomposeArray(constellation);
    i = single(i);
    q = single(q);
end

function [subcarrier, evm] = FetchEvmVsSubcarrier(app, selectorString, timeout)
    [-, evmNet] = app.nr.ModAcc.Results.FetchRmsEvmPerSubcarrierMeanTrace(selectorString, timeout, []);
    evm = evmNet.GetRawData();
    evm = single(evm);
    subcarrier = (0:single(evmNet.SampleCount) - 1) * evmNet.Timing.SampleInterval.TotalSeconds;
end

function [symbol, evm] = FetchEvmVsSymbol(app, selectorString, timeout, [])
    [-, evmNet] = app.nr.ModAcc.Results.FetchRmsEvmPerSymbolMeanTrace(selectorString, timeout, []);
    evm = evmNet.GetRawData();
    evm = single(evm);
    symbol = (0:single(evmNet.SampleCount) - 1) * evmNet.Timing.SampleInterval.TotalSeconds;
end

function UpdateDisplay(app, resultName, vsaMeasurementTimeout)
    selectorString = strcat('result:', resultName);
    if strcmp(resultName, 'preDpd')
        resultIndex = 0;
        color = 'b';
    else
        color = 'r';
        resultIndex = 1;
    end
    dot = strcat(color, '.');
    line = strcat(color, '-');
    try
        % Fetch and Plot ACP %
        [df, spectrum] = app.FetchACP(selectorString, vsaMeasurementTimeout);
        plot(app.Graph0, df, spectrum, line);
        axis(app.Graph0, 'manual');
    end
end
    
```

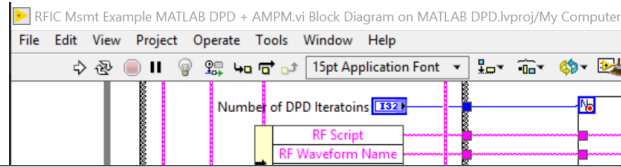
The interface includes several plots:

- Constellation Plot:** Shows four clusters of blue data points in the complex plane, with a red dot at the center of each cluster.
- M-Gain Plot:** A waterfall plot showing signal gain across a frequency spectrum from 2.1 to 2.4 GHz.
- Gain vs Input Power Plot:** A scatter plot showing Gain (dBm) on the y-axis (50 to 53) versus Input Power (dBm) on the x-axis (-30 to -5). It features a dense cloud of blue points and a red line indicating a trend.

At the bottom center, there is a prominent **Run** button.

Measurement	Before DPD	After DPD
ACP NR Channel Power (d...	35.5436	35.2079
ACP NR Offset 1 Lower (dB)	-26.3277	-51.0464
ACP NR Offset 1 Upper (dB)	-28.5067	-51.5999
ACP NR Offset 2 Lower (dB)	-43.7083	-52.8022
ACP NR Offset 3 Upper (dB)	-48.8719	-53.6735
EVM (%)	5.9039	0.3752

Validation Engineer's View in LabVIEW



RFIC Msmt Example MATLAB DPD + AMPM.vi Front Panel on MATLAB DPD.lvproj/My Computer

Center Frequency (Hz) 2.150000G

RFSG RFSA

RFSA Resource Name VST_01

Reference Level (dBm) 45.0

External Attenuation (dB) 42.00

Trigger

Enable Trigger?

Digital IQ Power Edge

Digital Edge Source PXI_Trig0

Trigger Delay (s) 0.00

status code source

DPD SGNR AM/PM

NanoSemi Linearizer

Enable DPD?

DPD Configuration

DPD Level Level 2

rho 0.1

Training Samples 25k

Acquisition Time (s) 200.00u

Number of DPD Iterations 3

Post DPD Results

Constellation Plot 0

EVM vs Symbol Plot 0

ACP Plot 0

EVM vs Subcarrier Plot 0

Parameter	Value
RMS EVM Mean (%)	0.362867
NR Offset 1 Lower Relative Power (dB)	-51.10
NR Offset 1 Upper Relative Power (dB)	-51.61
NR Offset 2 Lower Relative Power (dB)	-52.73
NR Offset 2 Upper Relative Power (dB)	-53.57

fault DPD configuration, sets the appropriate fields, then resets DPD training;

MATLAB script

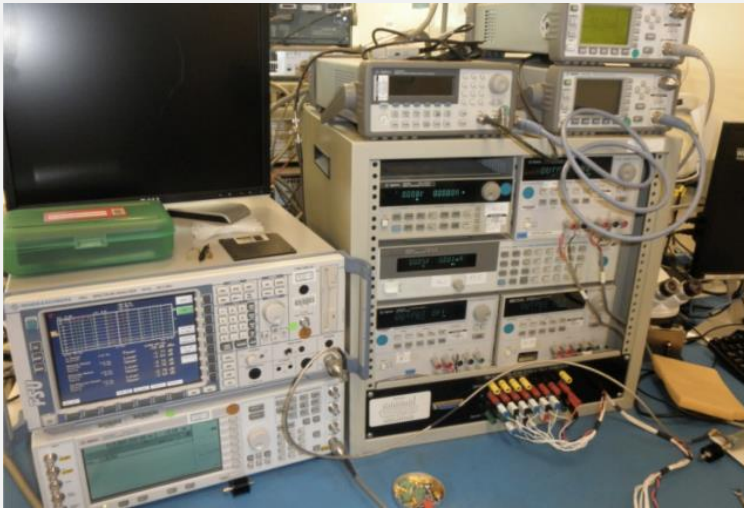
```

cd(applicationDirectory)
if ~libloaded('nstdpd')
loadlibrary('C:\NanoSemi\bin\nstdpd.dll', 'C:\NanoSemi\inc\nstdpd.h')
end
[error, dpdConfig] = calllib('nstdpd', 'nstdpd_get_default_config', []);
if ~strcmp(error, 'NST_SUCCESS')
error('Error occurred getting DPD default configuration');
end
dpdConfig.lv1 = dpdLevel;
dpdConfig.rho = dpdRho;
dpdConfig.training_samples = dpdTrainingSamples;
dpdConfig.abs_vsg_max = 1.01;
dpdConfig.f_sample = referenceWaveformSampleRate / 1e6;
error = calllib('nstdpd', 'nstdpd_reset_training', dpdConfig);
if ~strcmp(error, 'NST_SUCCESS')
error('Error occurred resetting DPD training');
end
    
```

error out

Two Distinct Approaches to PA Characterization

Traditional Approach



- **Separate** workflow for design and validation
- **Different** waveforms, PA models, analysis algorithm
- **Expensive**, **large** footprint, **poor** synchronization

Platform-Based Approach

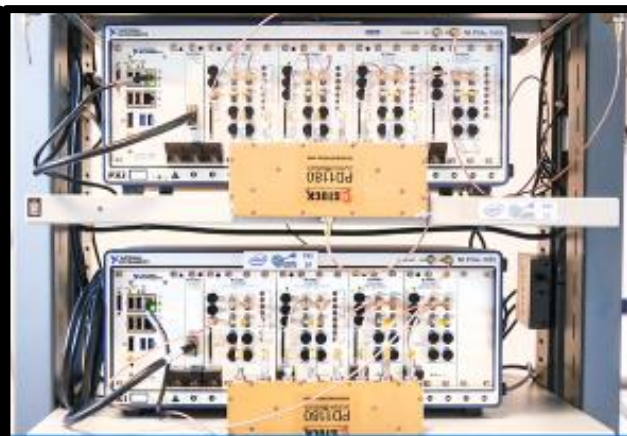


- **Integrated** workflow for design and validation
- **Same** waveforms, PA models, analysis algorithm
- **Modular**, **small** footprint, **sub-nanosecond** synchronization

Ultra High Band 5G FEM 3.3 – 4.2 GHz

400 MHz bandwidth

Rapidly tested with a wide variety of waveforms



5x faster test times

Reduced tester footprint by 50%

Saved Several Million \$\$\$

"The wide bandwidth, excellent RF performance, and the flexibility of NI's PXI test system were critical in helping us introduce the industry's first commercially available 5G FEM. Qorvo's focus on innovation was clearly demonstrated at the 20th GTI Workshop in London."

—Paul Cooper, Director of Carrier Liaison and Standards



ni.com

"The measurement speed of PXI was very attractive to us. In fact, the VST's measurement speed was about 5 times faster than our previous test equipment. This has allowed us to cut the characterization time for a typical LTE modem from one week to less than 2 days... With the additional testing that we were able to perform using PXI, we estimate that we have saved several million of dollars."

—Eike Ruttkowski, Head of RF Cellular Hardware



ni.com



Sub-6 GHz New Radio Sky5 (3.3 – 5.0 GHz)

200 MHz bandwidth

Tested with the PXIe-5840 VST



"We were able to reduce manufacturing test time of Power Amp (PA) by 5 times compared to existing test system by using NI VST to implement power servoing on FPGA level."

—New Product Introduction (NPI) Team, Broadcom



ni.com

"Skyworks is pleased to be utilizing NI's RF VST to validate performance of our Sky5™ solutions for 5G NR applications. Using NI's PXI platform, we are able to validate key performance benchmarks."

—Kevin Walsh, Senior Director of Mobile Marketing for Skyworks



ni.com

Qualcomm UK Uses MATLAB to Develop 5G RF Front-End Components and Algorithms

Challenge

10x more waveform combinations in 5G than in LTE, making device validation much more complex and time-consuming

Solution

Use MATLAB to simulate hardware-accurate Tx and Rx paths to predict system performance and optimize design parameters.

Results

- Fully model RF transceiver and components
- Securely release sensitive IP
- Eliminate the cost of developing separate test suites



Qualcomm 5G RF front end prototype
MATLAB EXPO 2019

“We use MATLAB models to optimize and verify the 5G RF front end through all phases of development.”

Sean Lynch
Qualcomm UK, Ltd.

NanoSemi Improves System Efficiency for 5G and Other RF Products

Challenge

Accelerate design and verification of RF power amplifier linearization algorithms used in 5G and Wi-Fi 6 devices

Solution

Use MATLAB to characterize amplifier performance, develop predistortion and machine learning algorithms, and automate standard-compliant test procedures

Results

- Development time reduced by 50%
- Iterative verification process accelerated
- Early customer validation enabled



NanoSemi linearization IP development and verification using MATLAB.

“With MATLAB, our team can deliver leading-edge IP faster, enabling our customers to increase bandwidth, push modulation rates higher, and reduce power consumption.”

Nick Karter
NanoSemi

Wrap up

- How MATLAB and Simulink can be used in a wireless system design workflow
- Wireless Scenario Simulation
- End-to-end Simulation of mmWave Communication Systems with Hybrid Beamforming
- Developing Power Amplifier models and DPD algorithms in MATLAB
- Use of National Instruments PXI for PA characterization with DPD

Learn More

- Where can you get more information about MathWorks tools for wireless system modelling?
- [MATLAB and Simulink for 5G Development](#)
- White paper: [RF PA and DPD linearization using MATLAB and Simulink](#)
- White paper: [Hybrid Beamforming for 5G Systems](#)