MATLAB EXPO

FPGA-Based Implementation of Beamforming Algorithms for Radar and Wireless Systems

Sumit Garg, MathWorks

Kishore Siddani, MathWorks

MathWorks®

MathWorks
@MathWorks

Share the EXPO experience
#MATLABEXPO

linkedin.com/in/
sumit-garg-689bb916

linkedin.com/in/
kishore-siddani/

Phased array systems are used in many applications

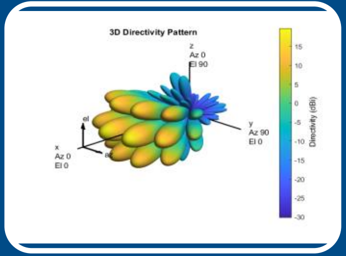
Multifunction Radars

Wireless Communications

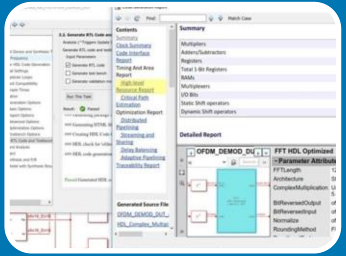
Satellite Communications

Acoustics

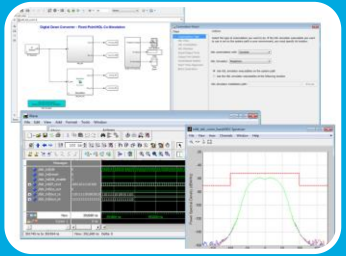
Key Takeaways



Beamforming for interference mitigation

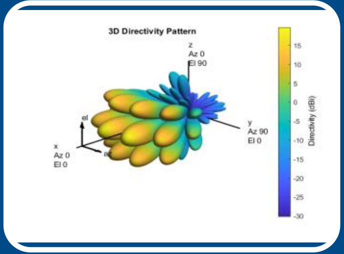


HDL Code Generation for Beamforming algorithms

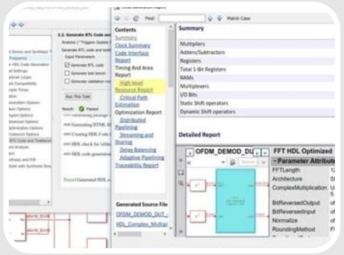


Integrated Verification of HDL Code

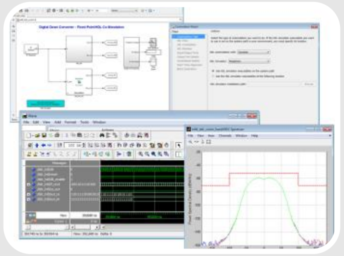
Key Takeaways



Beamforming for interference mitigation

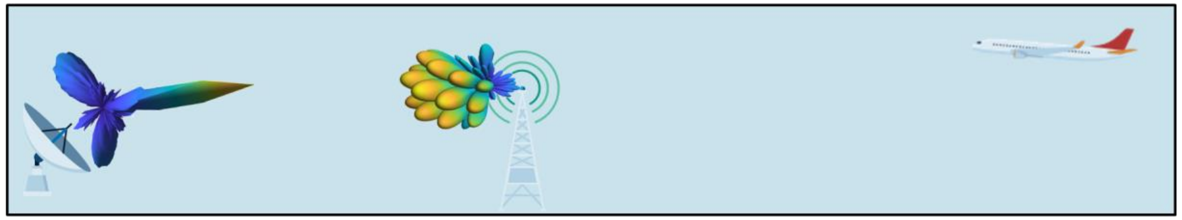


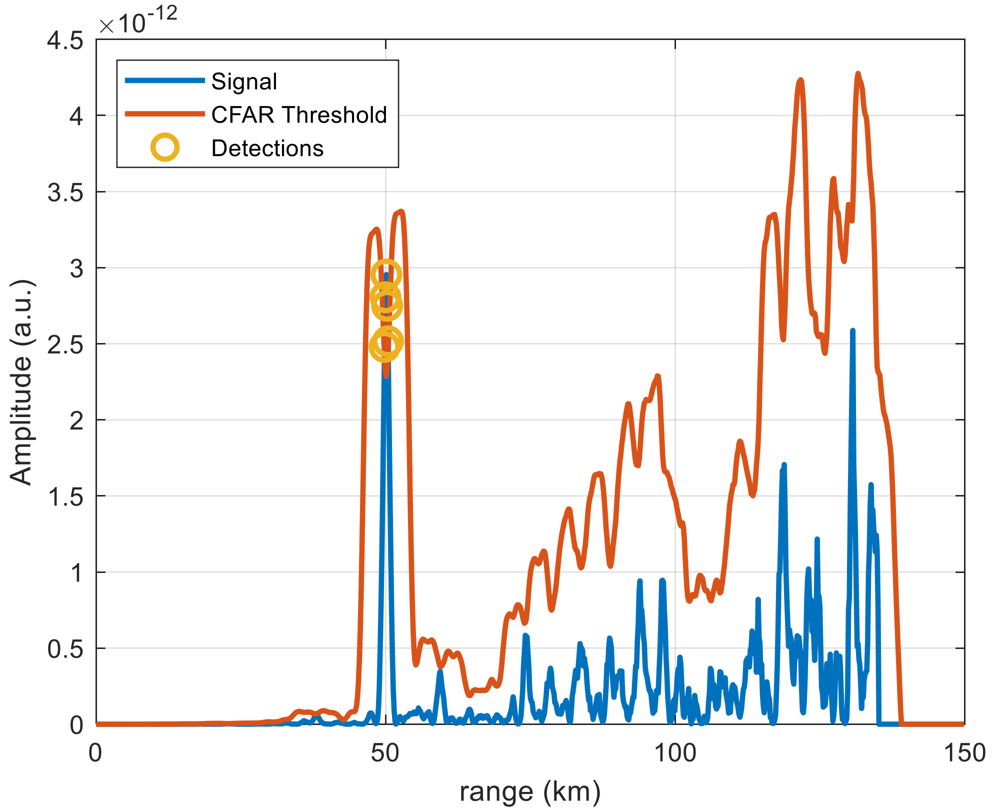
HDL Code Generation for Beamforming algorithms

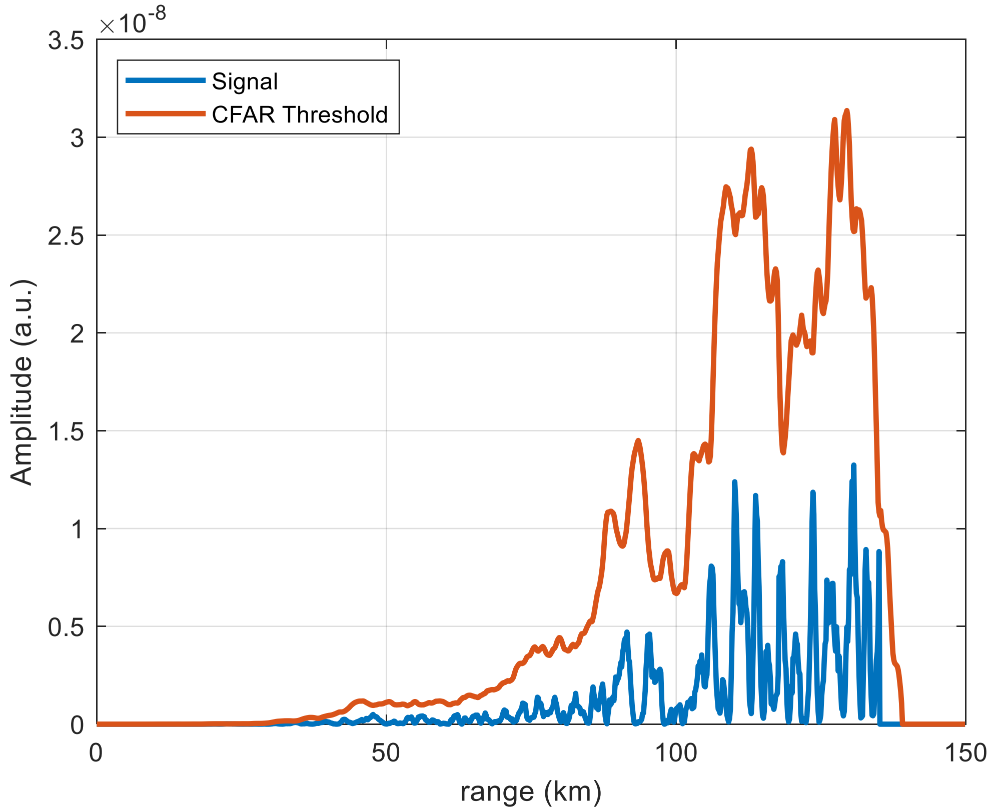


Integrated Verification of HDL Code

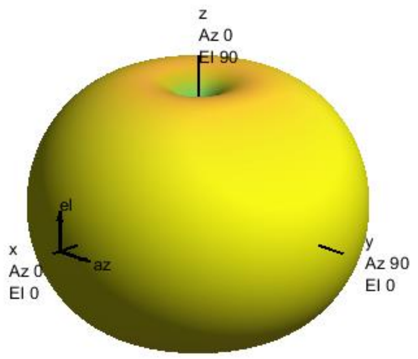
Model interference between airport surveillance radar and 5G





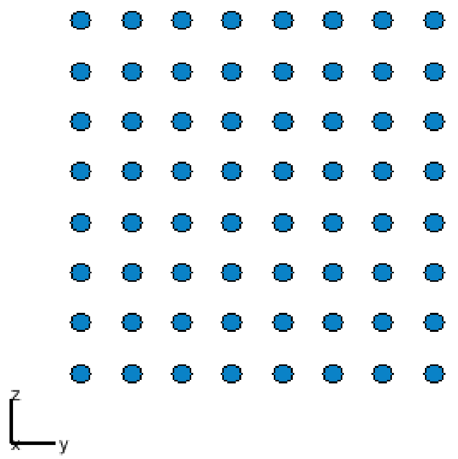


Total array pattern can be computed from pattern multiplication



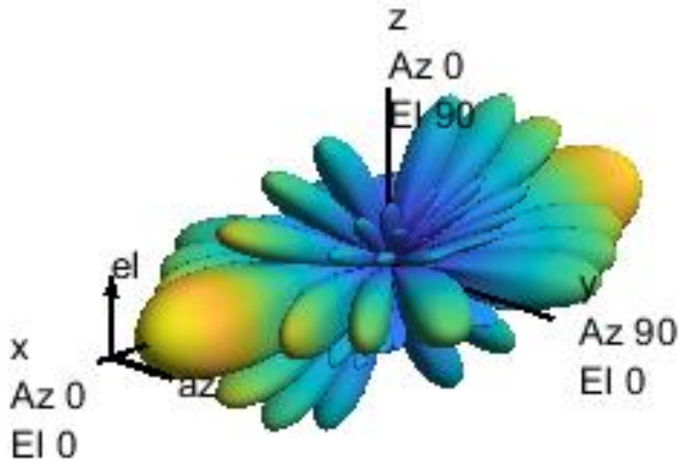
×

Array Geometry

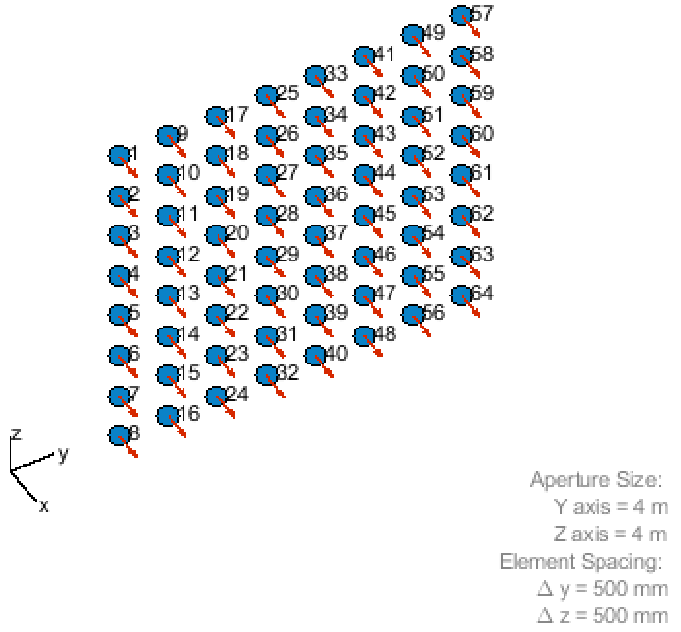


Aperture Size:
Y axis = 4 m
Z axis = 4 m
Element Spacing:
Δ y = 500 mm
Δ z = 500 mm

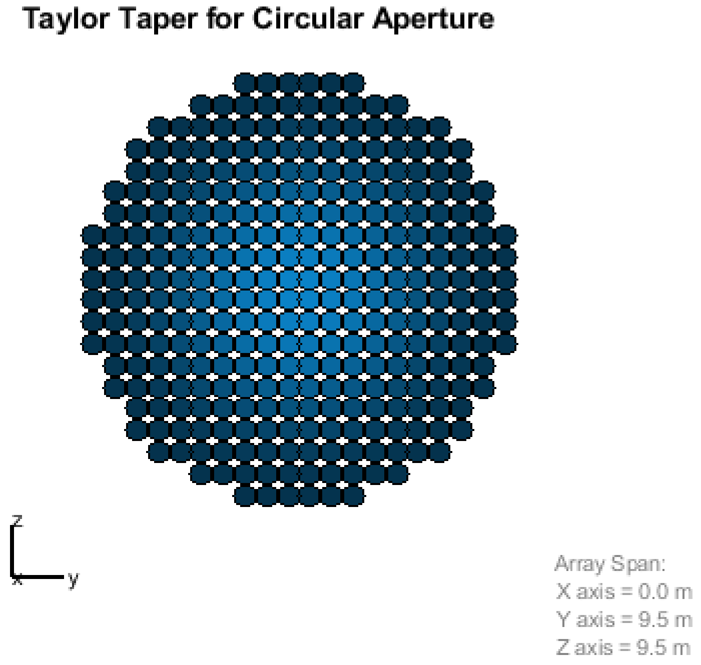
=



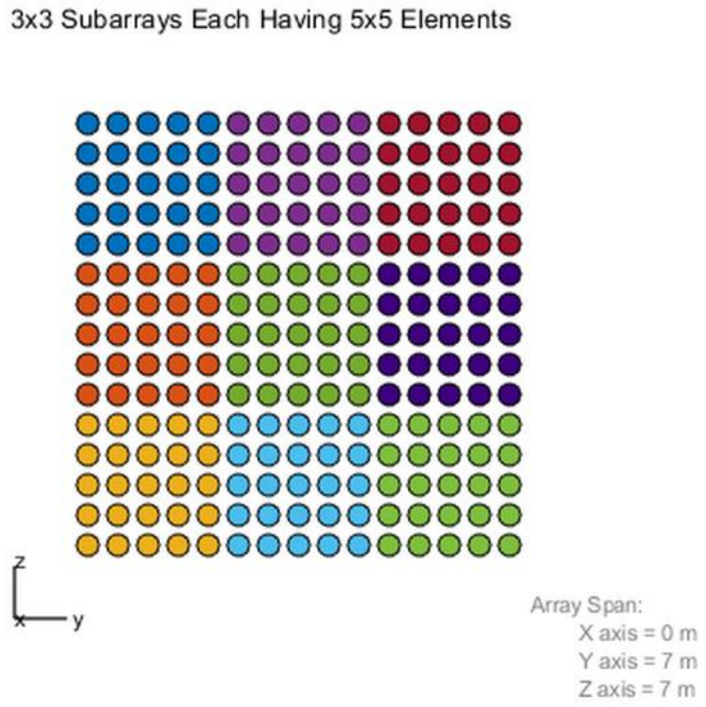
There are many parameters needed to model an array



Element position and normal



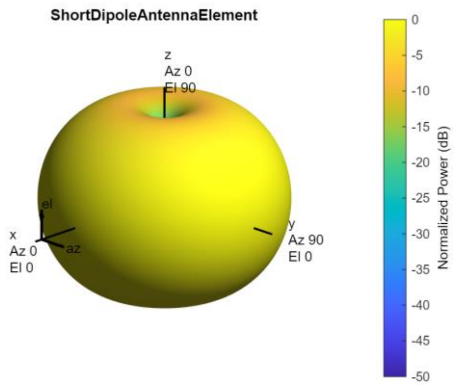
Element taper

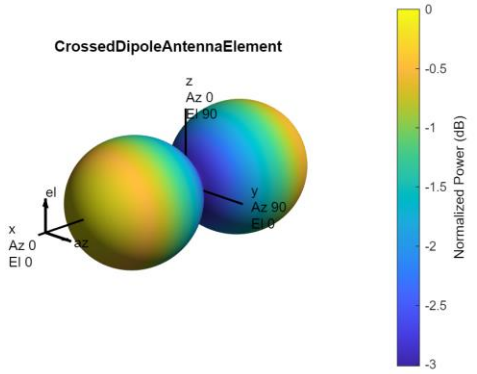


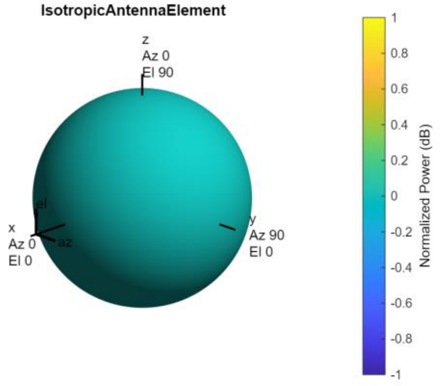
Subarray architecture

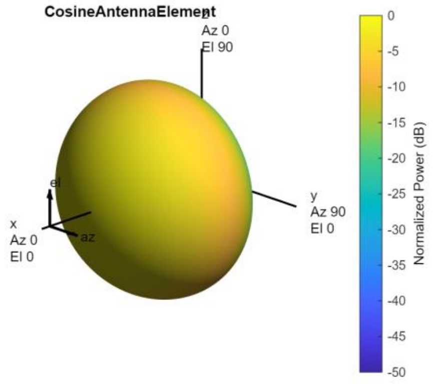
There are multiple mathematical patterns to get started with

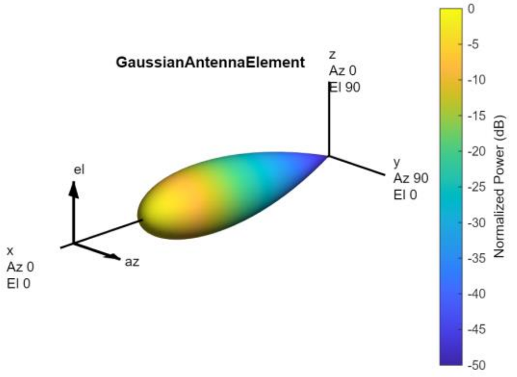
No Polarization	Polarized
Isotropic	Short dipole
Cosine	Crossed dipole
Gaussian	
Cardioid	
Sinc	

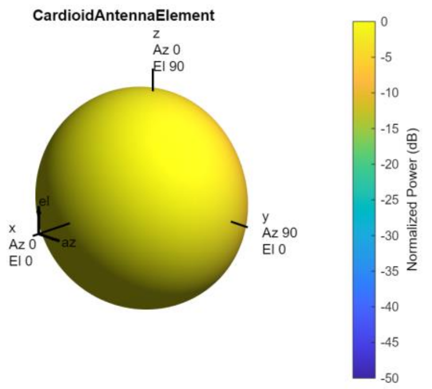


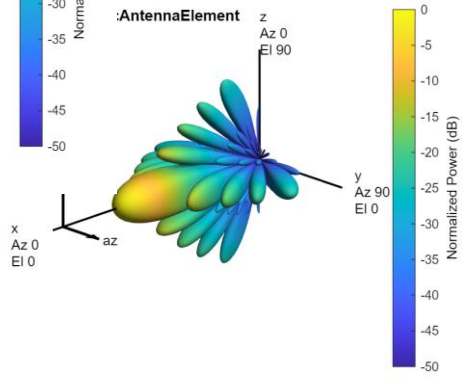






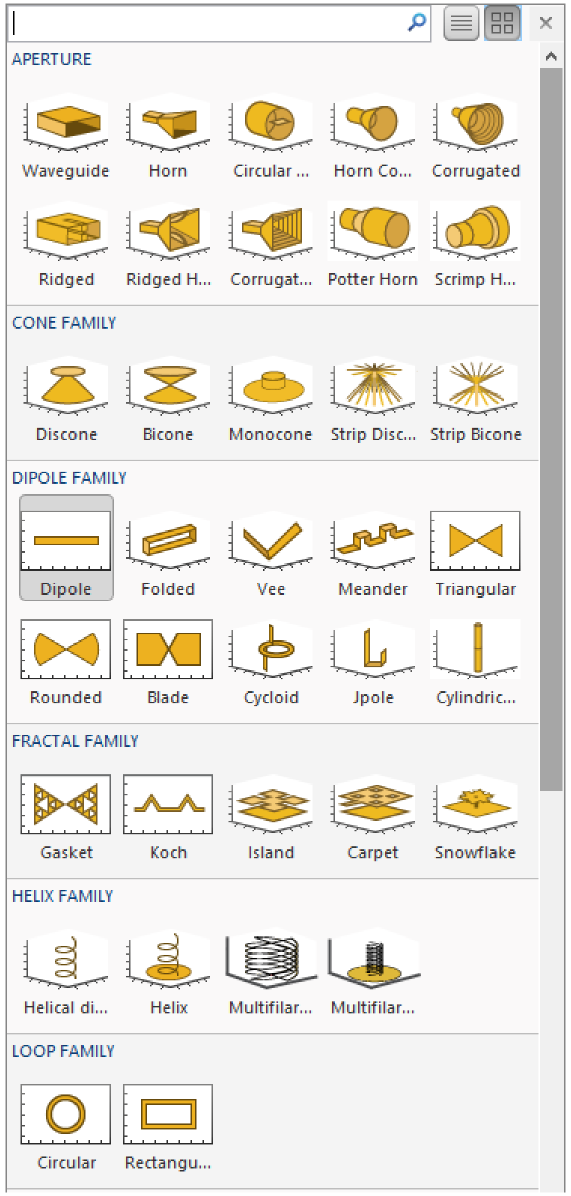


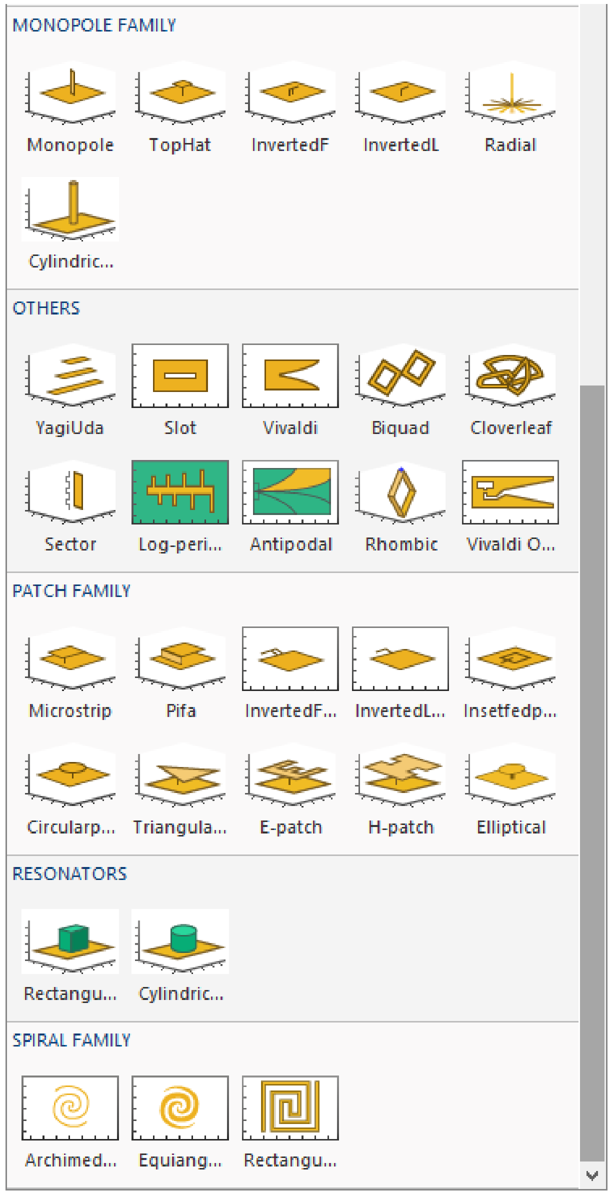




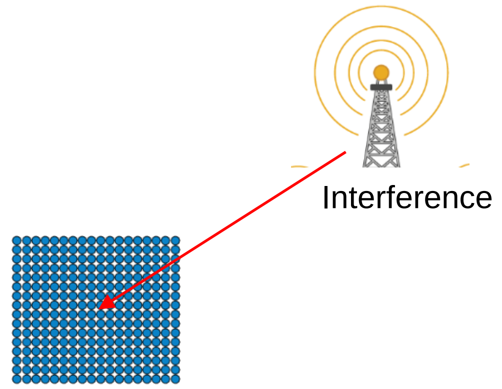
Antenna Toolbox provides many additional antenna elements

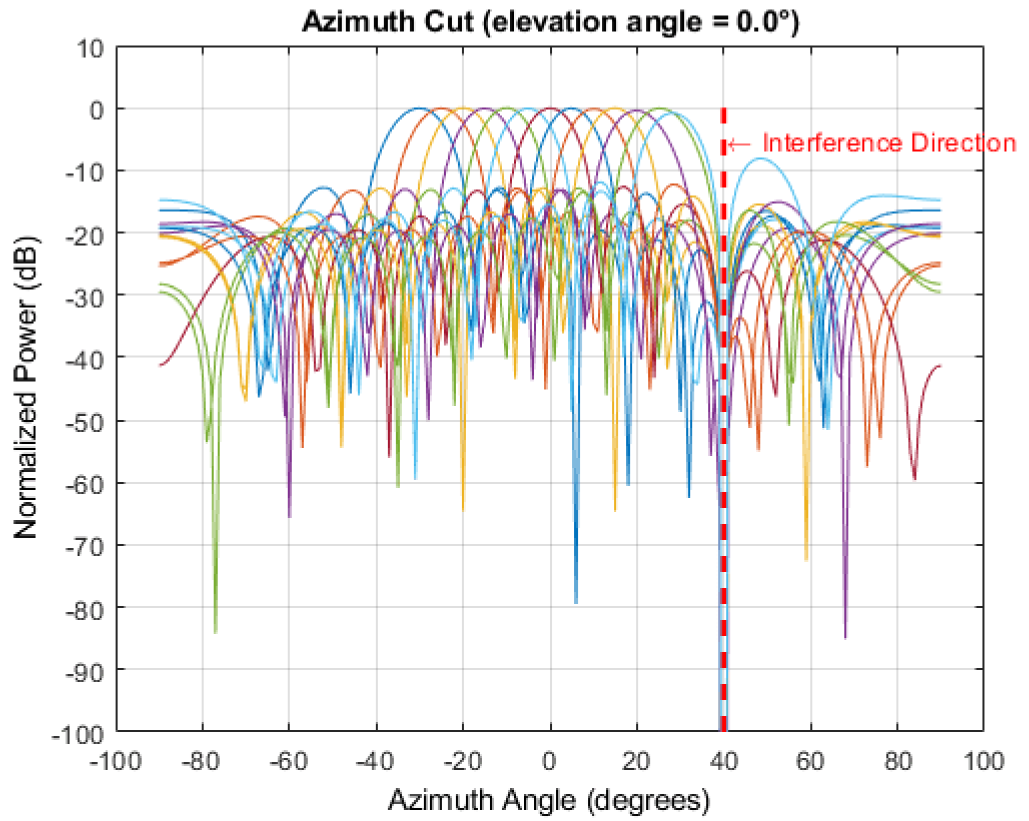
- Dipole and bowtie antennas
- Monopole antennas
- Patch antennas
- Spiral and loop antennas
- Slot antennas
- Helix antennas
- Fractal antennas
- Waveguide antennas
- Horn and cone antennas
- Other common antennas
- Backing structures
- Import custom antenna pattern

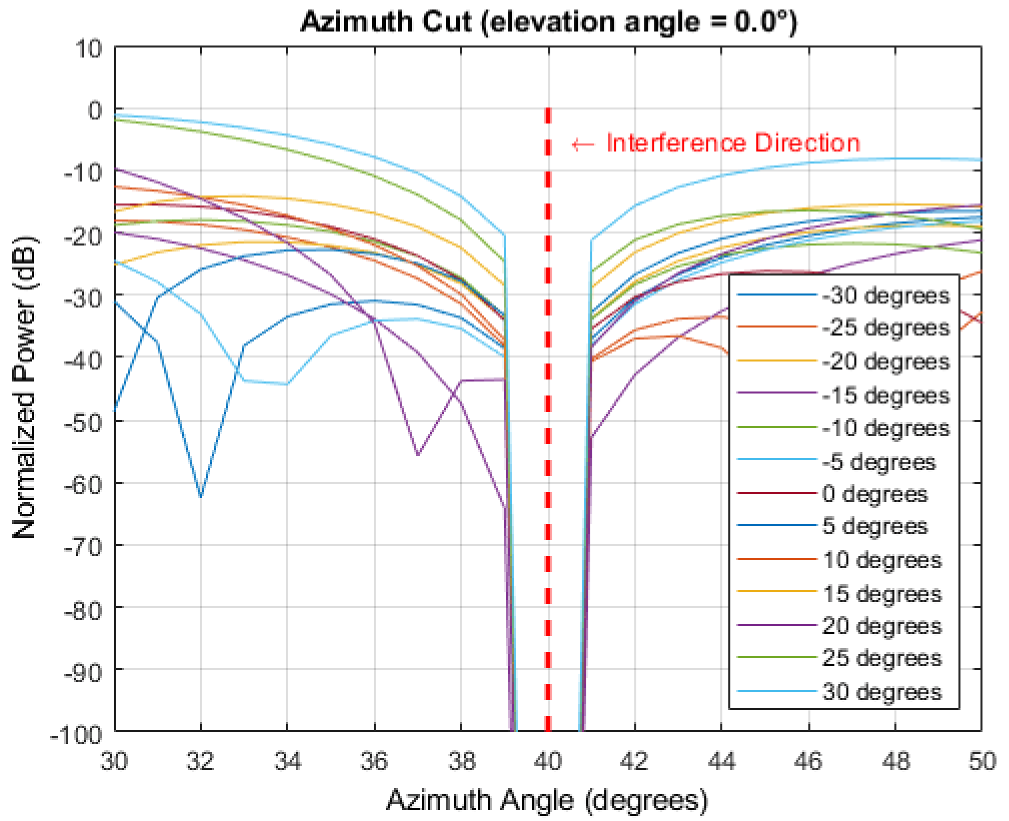




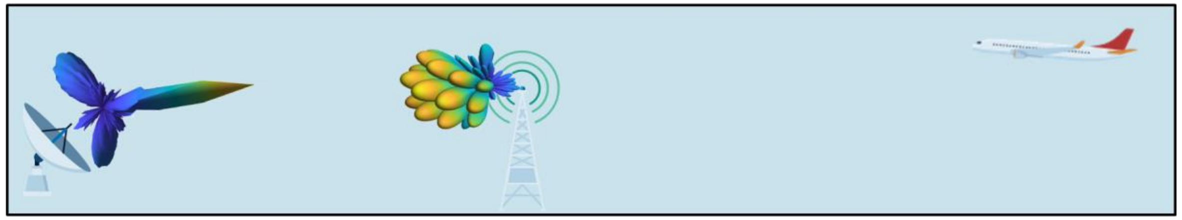
Null out the interference with beamforming

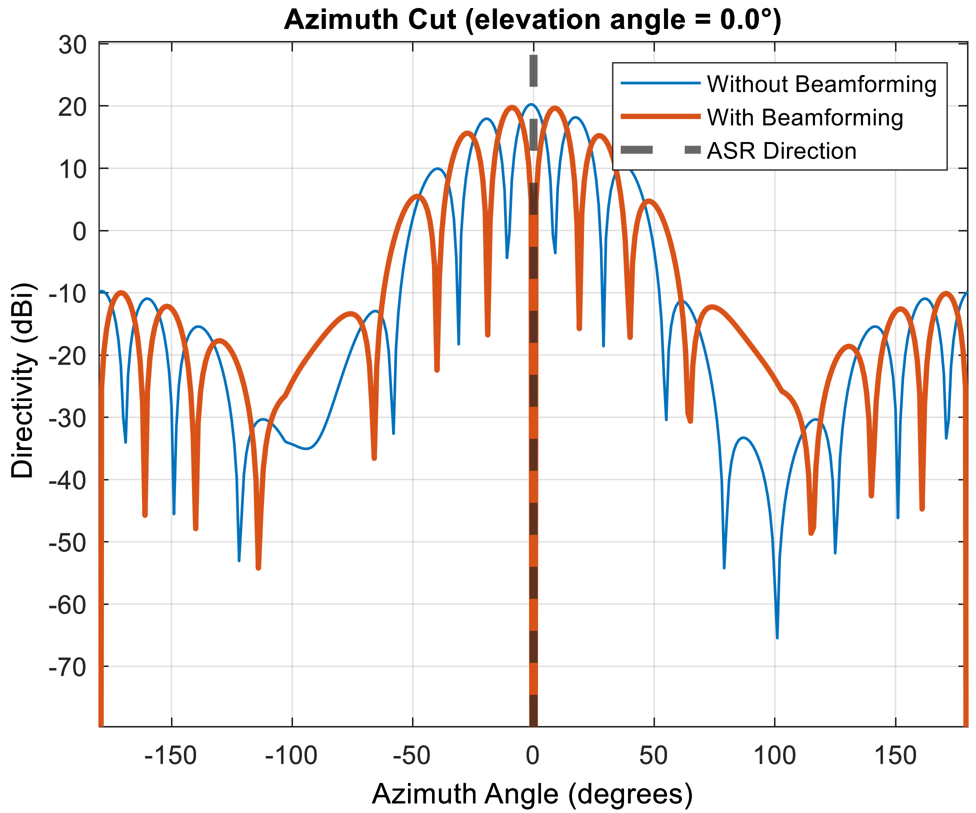


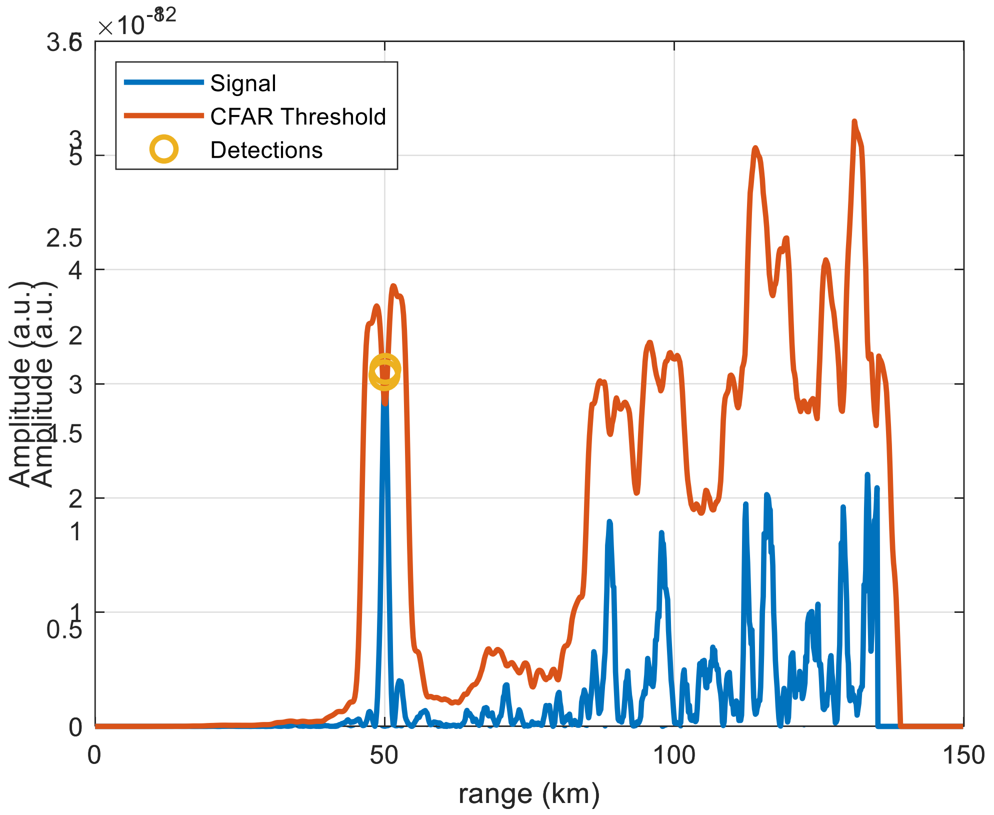




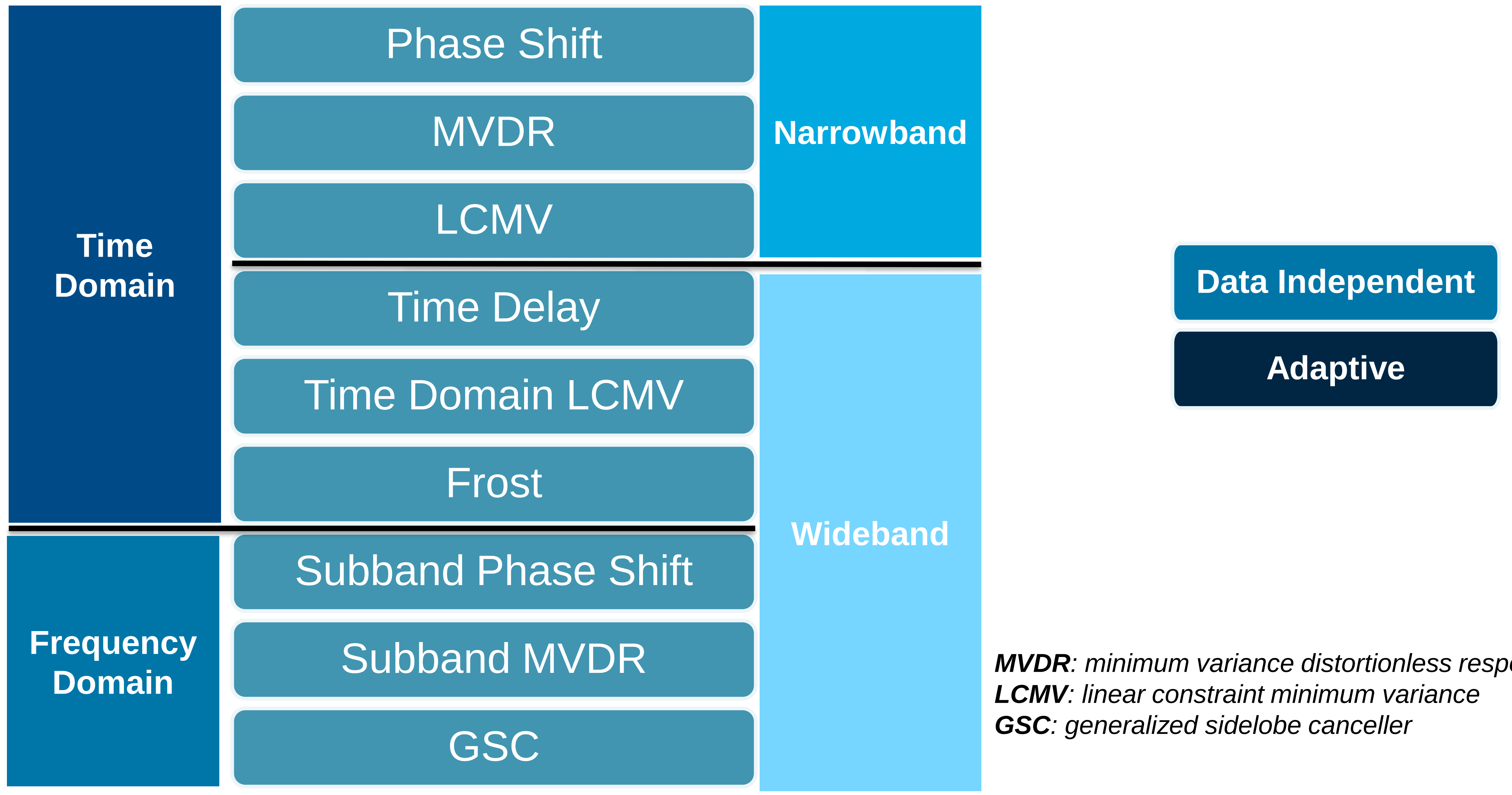
Reduce the interference by nulling the base station beam pattern toward the radar





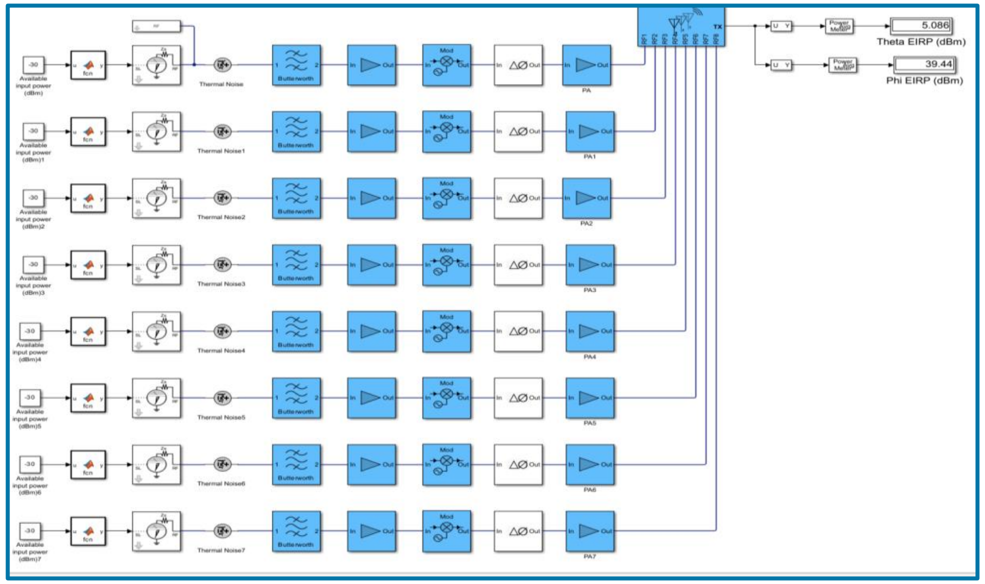


Beamformers can be categorized in many ways

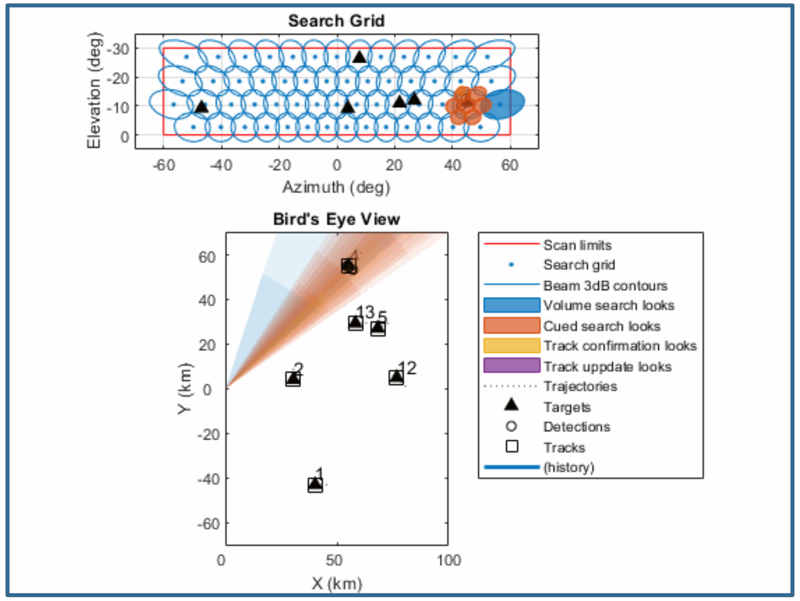


MVDR: *minimum variance distortionless response*
LCMV: *linear constraint minimum variance*
GSC: *generalized sidelobe canceller*

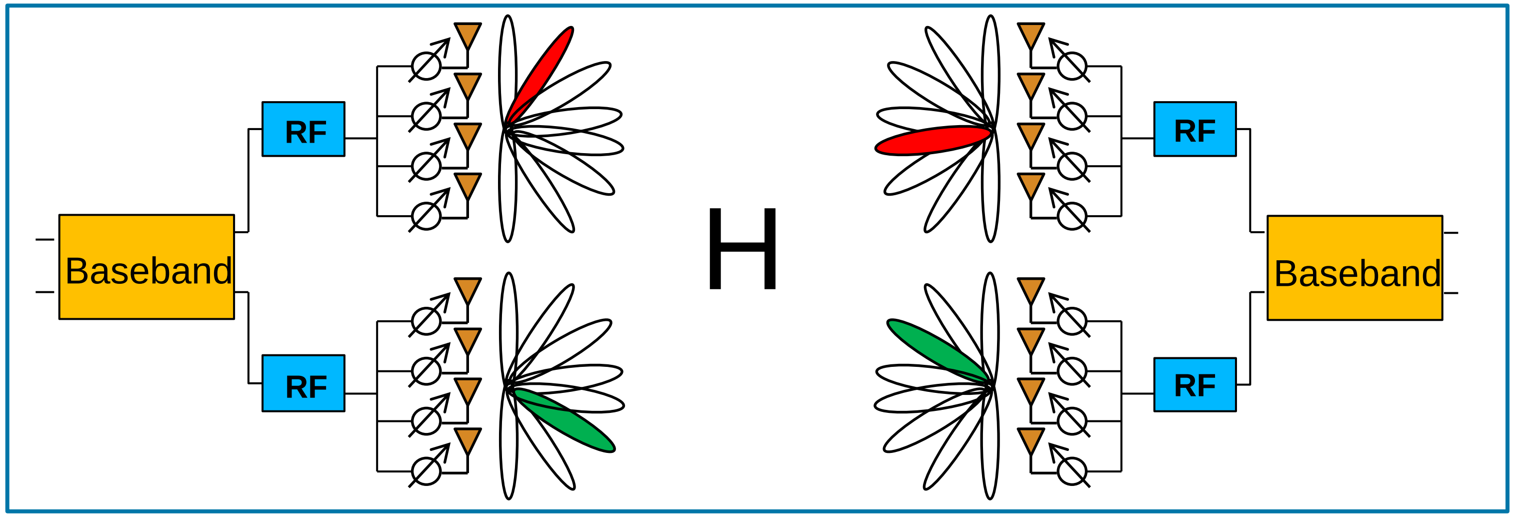
There are many beamforming options with phased arrays



RF Beamforming

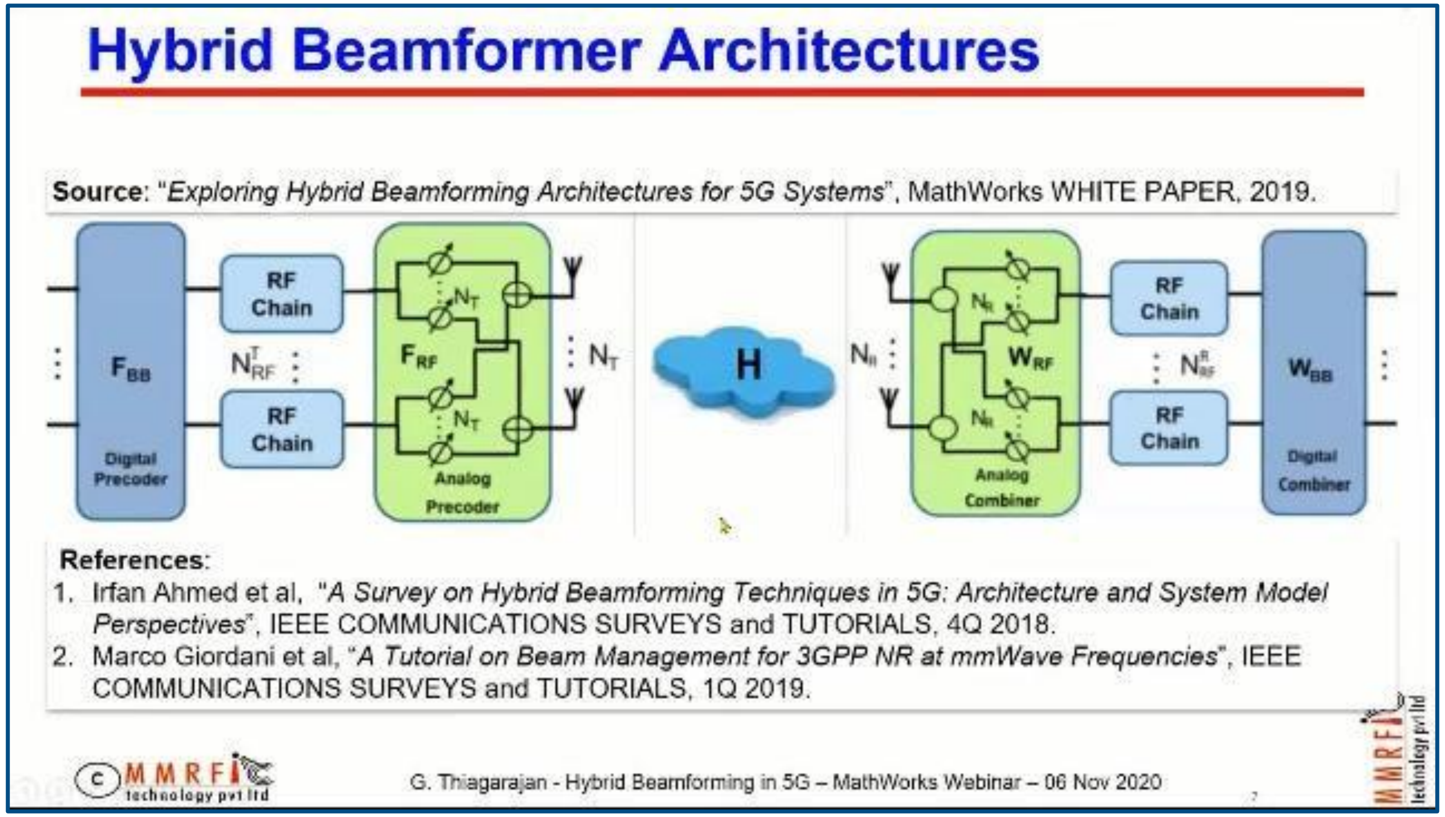


Digital Beamforming



Hybrid Beamforming

MMRFIC Implements a 5G Massive MIMO Array with Hybrid Beamforming



Hybrid Beamformer Architectures

Source: "*Exploring Hybrid Beamforming Architectures for 5G Systems*", MathWorks WHITE PAPER, 2019.

References:
1. Irfan Ahmed et al, "*A Survey on Hybrid Beamforming Techniques in 5G: Architecture and System Model Perspectives*", IEEE COMMUNICATIONS SURVEYS and TUTORIALS, 4Q 2018.
2. Marco Giordani et al, "*A Tutorial on Beam Management for 3GPP NR at mmWave Frequencies*", IEEE COMMUNICATIONS SURVEYS and TUTORIALS, 1Q 2019.

© MMRFIC technology pvt ltd

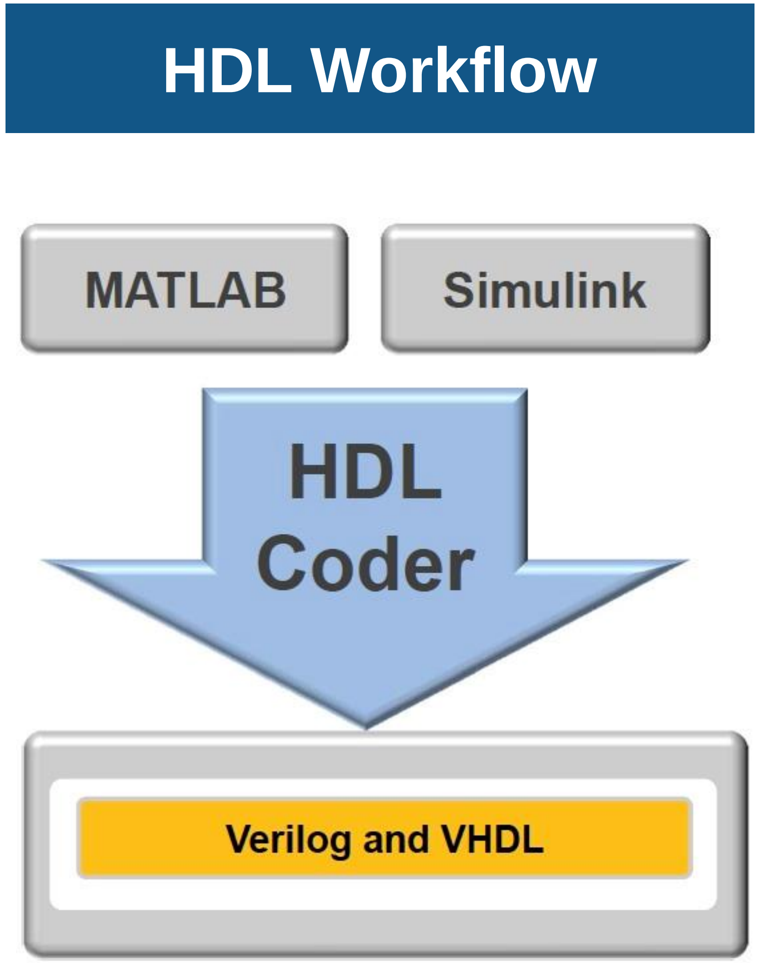
G. Thiagarajan - Hybrid Beamforming in 5G – MathWorks Webinar – 06 Nov 2020

"Hybrid beamforming system design for 5G massive MIMO arrays using MATLAB, Phased Array System Toolbox, and 5G Toolbox helped us in evaluating various hardware options as well as their performance in realistic 5G scenarios."
- Ganesan Thiagarajan, C.T.O., MMRFIC Technology Private Limited, India

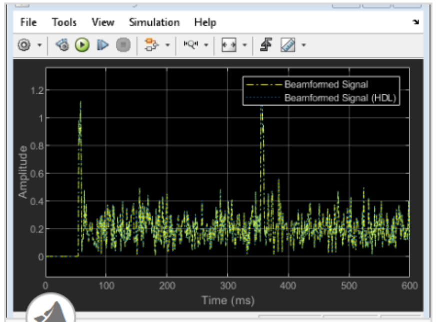
Hybrid beamforming system structure: transmitter, channel, and receiver.

Link to case study

Generate HDL for Beamforming Algorithms

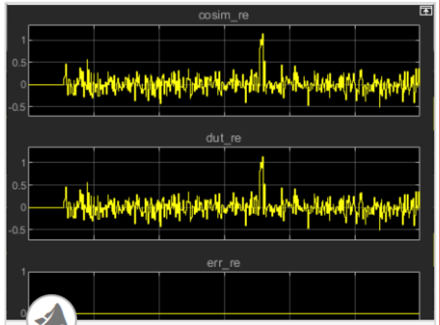


- Workflow illustration
  - Algorithm modeling
  - HDL code generation
  - Testing and verification



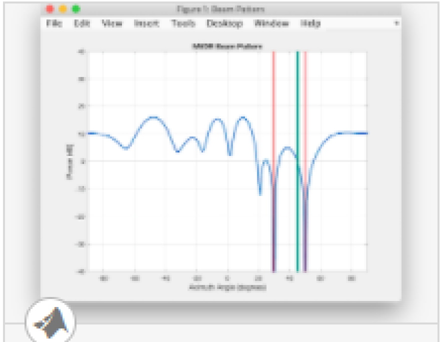
FPGA Based Beamforming in Simulink: Part 1 - Algorithm Design

This tutorial is the first of a two-part series that will guide you through how to develop a beamformer in Simulink suitable for implementation



FPGA Based Beamforming in Simulink: Part 2 - Code Generation

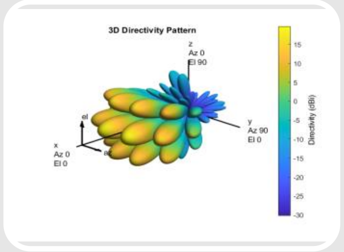
This tutorial is the second of a two-part series that will guide you through the steps to setup a Simulink implementation model to



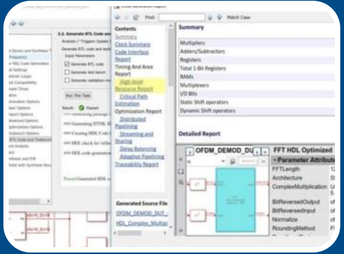
Fixed-Point HDL-Optimized Minimum-Variance Distortionless-Response...

Implement a fixed-point HDL-optimized minimum-variance distortionless-response beamformer.

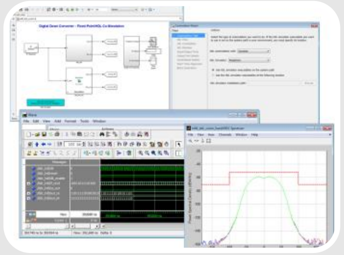
Key Takeaways



Beamforming for interference mitigation

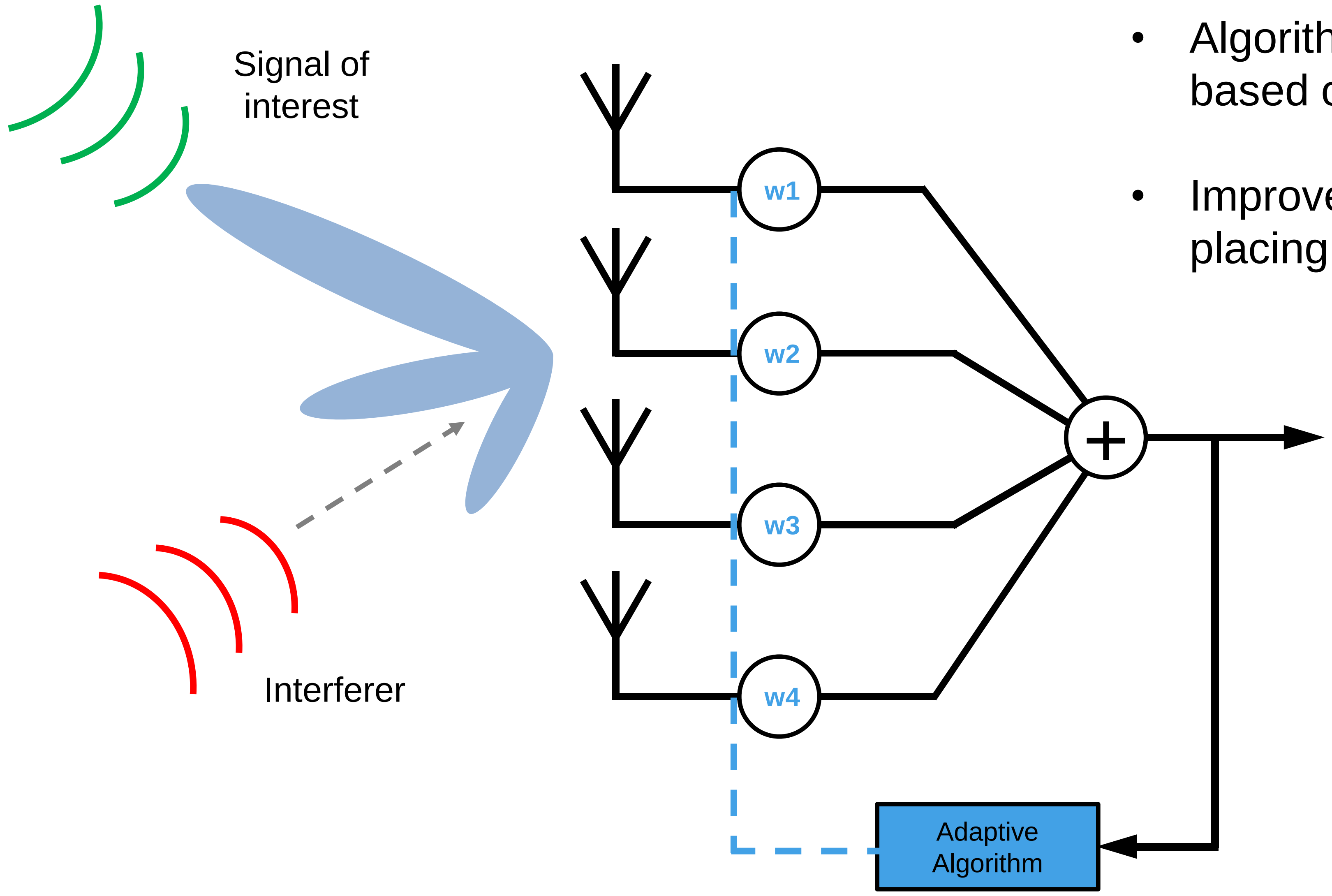


HDL Code Generation for Beamforming algorithms

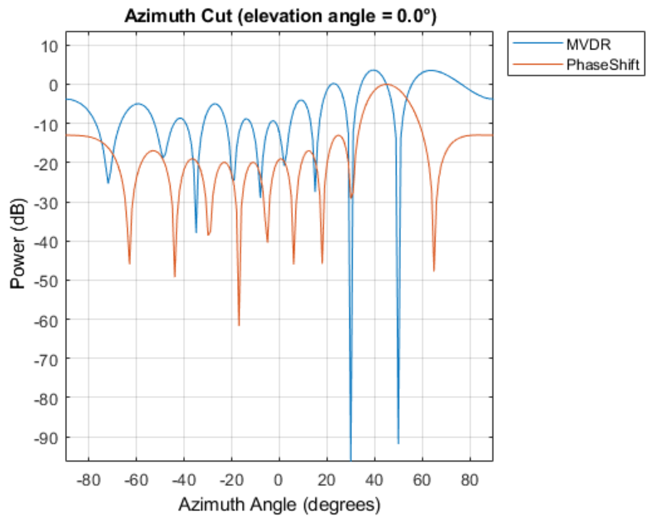


Integrated Verification of HDL Code

Adaptive Beamforming (**M**inimum **V**ariance **D**istortionless **R**esponse!)

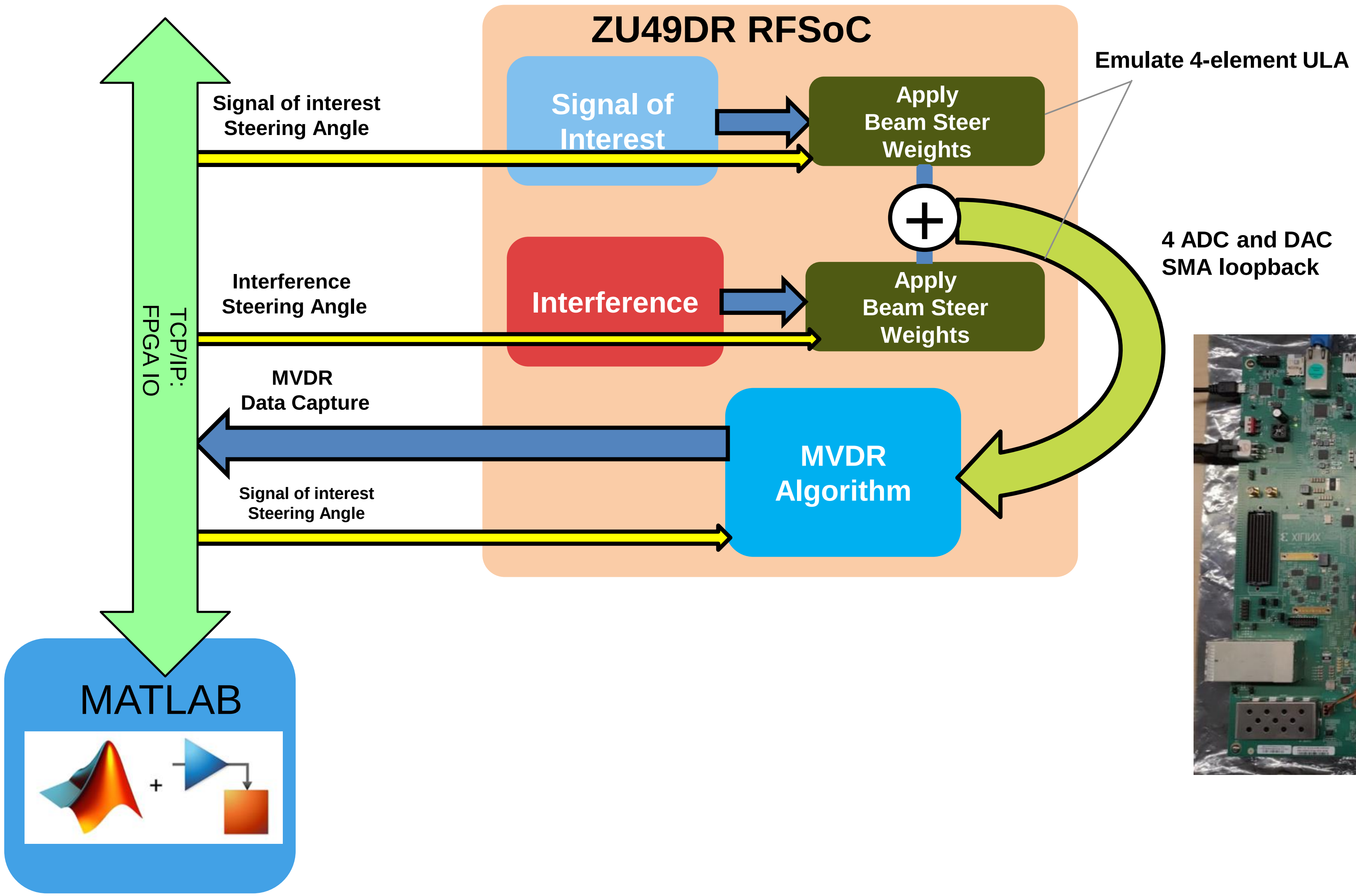


- Algorithm chooses optimal weights based on receive data statistics
- Improve SNR by automatically placing nulls at interference angles

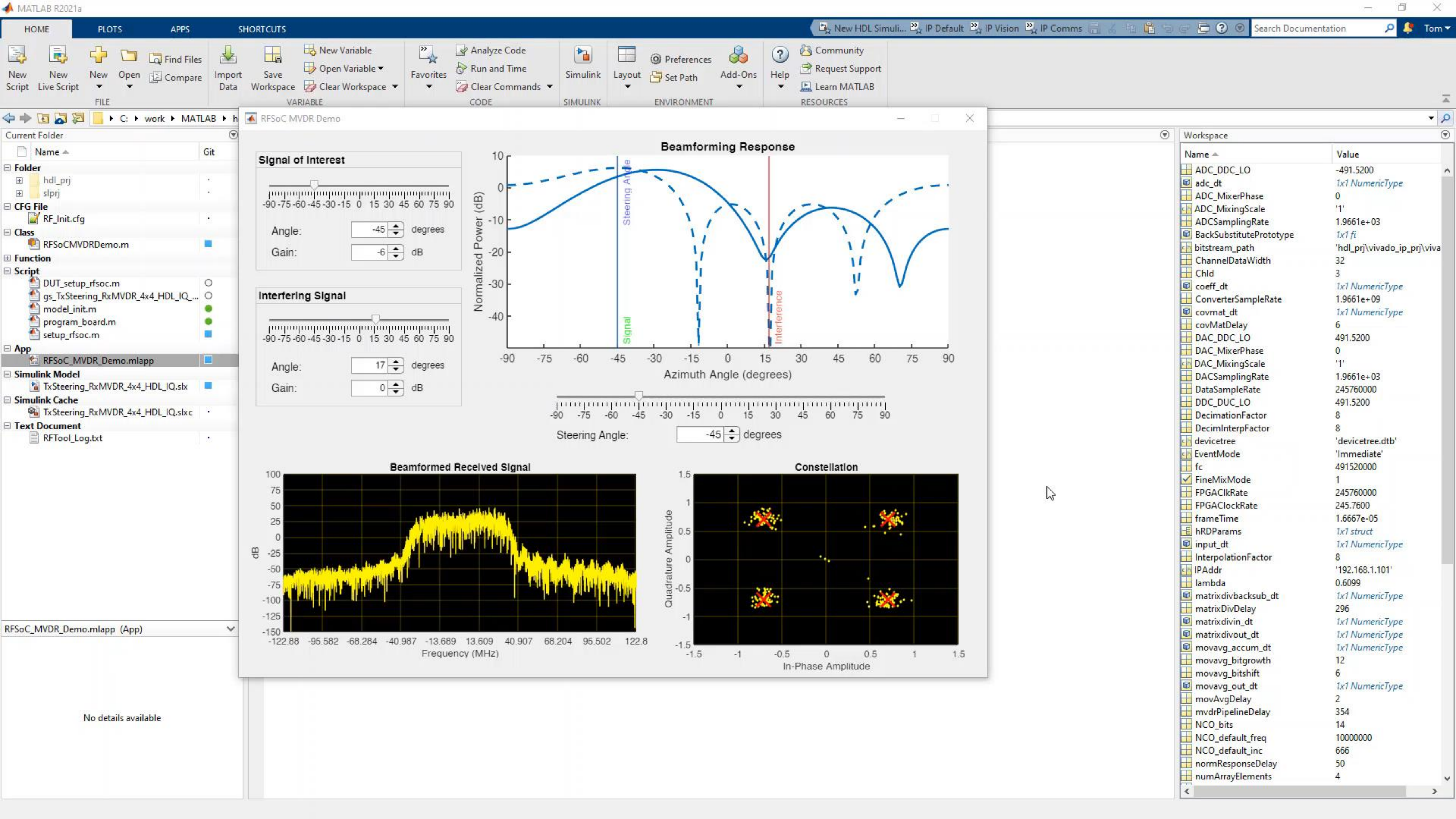


Beamforming Demonstration

Test Setup



4 ADC and DAC SMA loopback



Current Folder

Name	Git
Folder	
hdl_prj	·
slprj	·
CFG File	
RF_Init.cfg	·
Class	
RFSoCMVDRDemo.m	■
Function	
Script	
DUT_setup_rfsoc.m	○
gs_TxSteering_RxMVDR_4x4_HDL_IQ_...	○
model_init.m	●
program_board.m	●
setup_rfsoc.m	■
App	
RFSoC_MVDR_Demo.mlapp	■
Simulink Model	
TxSteering_RxMVDR_4x4_HDL_IQ.slx	■
Simulink Cache	
TxSteering_RxMVDR_4x4_HDL_IQ.slxc	·
Text Document	
RFTool_Log.txt	·

RFSoC_MVDR_Demo.mlapp (App)

No details available

Signal of Interest

Angle: -45 degrees

Gain: -6 dB

Interfering Signal

Angle: 17 degrees

Gain: 0 dB

Steering Angle: -45 degrees

Workspace

Name	Value
ADC_DDC_LO	-491.5200
adc_dt	1x1 NumericType
ADC_MixerPhase	0
ADC_MixingScale	'1'
ADCSamplingRate	1.9661e+03
BackSubstitutePrototype	1x1 fi
bitstream_path	'hdl_prj\vivado_ip_prj\viva
ChannelDataWidth	32
ChId	3
coeff_dt	1x1 NumericType
ConverterSampleRate	1.9661e+09
covmat_dt	1x1 NumericType
covMatDelay	6
DAC_DDC_LO	491.5200
DAC_MixerPhase	0
DAC_MixingScale	'1'
DACSamplingRate	1.9661e+03
DataSampleRate	245760000
DDC_DUC_LO	491.5200
DecimationFactor	8
DecimInterpFactor	8
devicetree	'devicetree.dtb'
EventMode	'Immediate'
fc	491520000
FineMixMode	1
FPGAClkRate	245760000
FPGAClockRate	245.7600
frameTime	1.6667e-05
hRDParams	1x1 struct
input_dt	1x1 NumericType
InterpolationFactor	8
IPAddr	'192.168.1.101'
lambda	0.6099
matrixdivbacksub_dt	1x1 NumericType
matrixDivDelay	296
matrixdivin_dt	1x1 NumericType
matrixdivout_dt	1x1 NumericType
movavg_accum_dt	1x1 NumericType
movavg_bitgrowth	12
movavg_bitshift	6
movavg_out_dt	1x1 NumericType
movAvgDelay	2
mvdrPipelineDelay	354
NCO_bits	14
NCO_default_freq	10000000
NCO_default_inc	666
normResponseDelay	50
numArrayElements	4

Live Demo available at our Technology Showcase booth

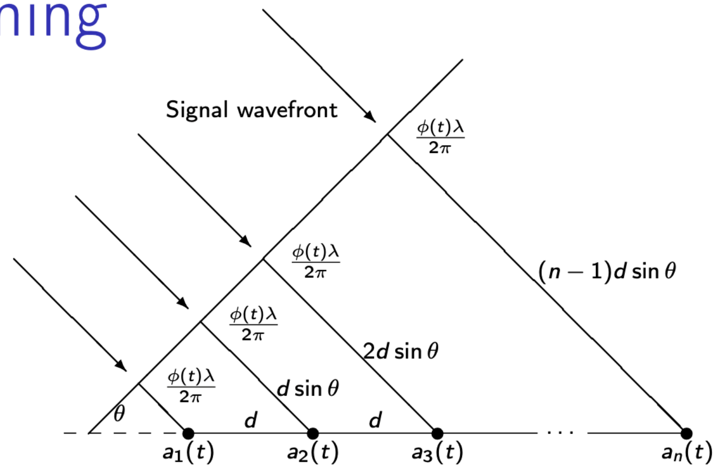
Implementing Adaptive Beamformer on RFSoC

- RFSoC Adaptive Beamformer with 4 channels
- Places nulls in interference locations and maximizes beam pattern for steering direction
- Interactively steer angles for interference and beam pattern at run time

MATLAB EXPO

Beamforming… Glance into Theory!

Beamforming


Signal wavefront
$\frac{\phi(t)\lambda}{2\pi}$
$\frac{\phi(t)\lambda}{2\pi}$
$\frac{\phi(t)\lambda}{2\pi}$
$\frac{\phi(t)\lambda}{2\pi}$
$(n-1)d\sin\theta$
$2d\sin\theta$
$d\sin\theta$
θ
d
d
$\cdots$
$a_1(t)$
$a_2(t)$
$a_3(t)$
$a_n(t)$

m samples, n antenna elements, $m \gg n$.

m-by-n data matrix A.

$a(t)$ is an n-by-1 column vector. $a(t)^H$ form the rows of A.

Unified notation

- A is the m-by-n data matrix
- $m \gg n$
- $A^H A$ is the n-by-n estimate of the covariance matrix
- $b = \begin{bmatrix} 1 \\ e^{(2\pi d/\lambda)\sin(\theta)i} \\ e^{2(2\pi d/\lambda)\sin(\theta)i} \\ \vdots \\ e^{(n-1)(2\pi d/\lambda)\sin(\theta)i} \end{bmatrix}$ is the steering vector

MVDR Beamformer Solution Steps in MATLAB

1) Form Covariance Matrix

$$A^H A$$

```
A'*A
```

2) Compute Weight Vector, Solve for 'x'

$$(A^H A)x = b$$

```
x = (A'*A)\b;
```

3) Normalize Response

$$w = \frac{x}{b^H x}$$

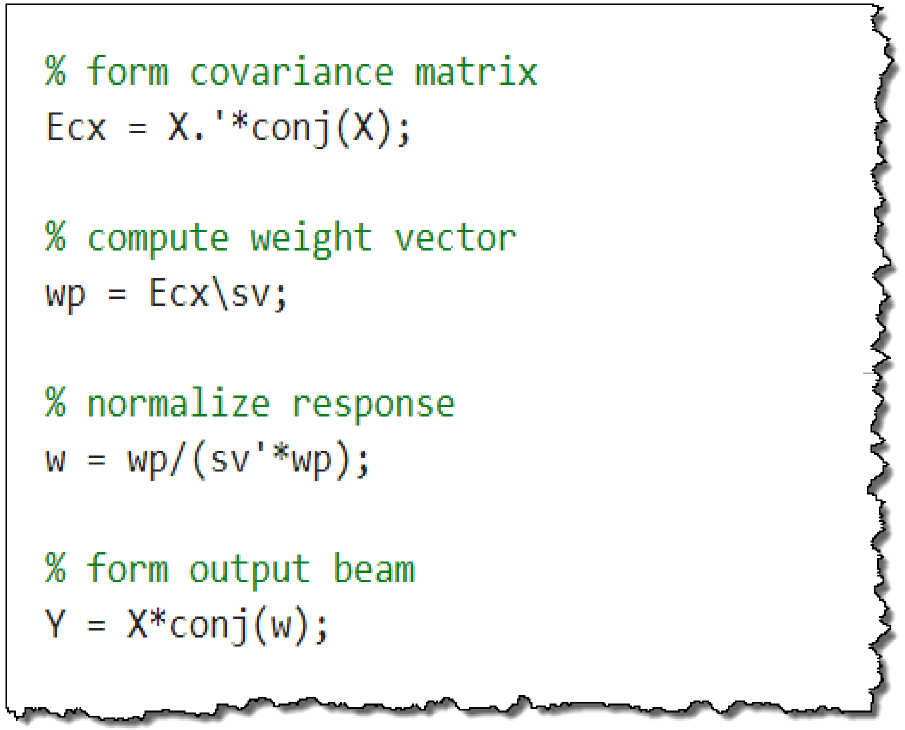
```
w = x/(b'*x);
```

4) Form Output Beam

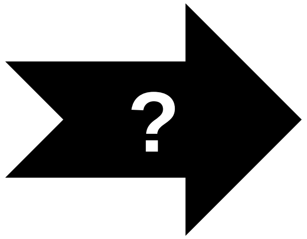
$$y = w^H a(t)$$

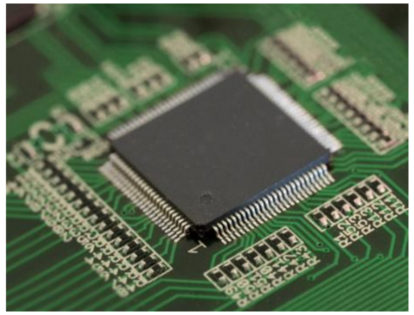
```
y = w'*a
```

How to Go from MATLAB Algorithm to HDL Implementation?



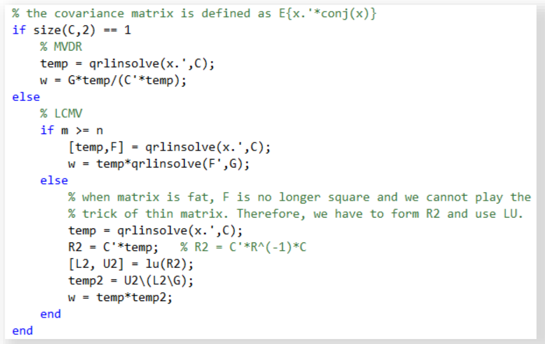
```
% form covariance matrix
Ecx = X.'*conj(X);

% compute weight vector
wp = Ecx\sv;

% normalize response
w = wp/(sv'*wp);

% form output beam
Y = X*conj(w);
```



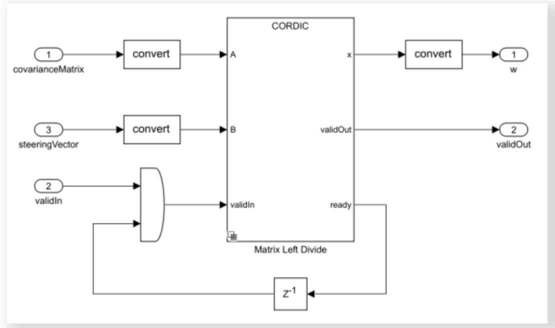


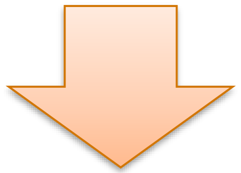
FPGA Implementation Challenges

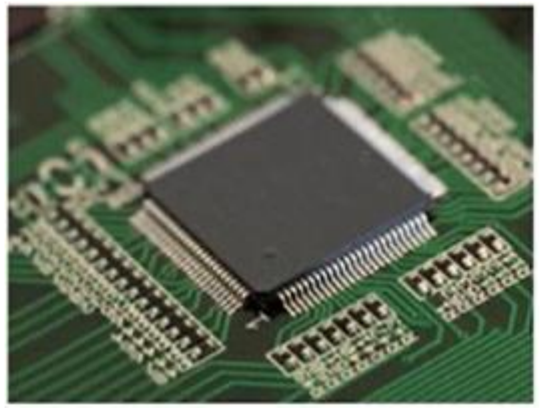
- Fixed-Point Math
- Performance vs Area tradeoffs
- Data Rate vs Clock Rate
- Project Timeline



```
% the covariance matrix is defined as E{x.'*conj(x)}
if size(C,2) == 1
    % MVDR
    temp = qrlinsolve(x.',C);
    w = G*temp/(C'*temp);
else
    % LCMV
    if m >= n
        [temp,F] = qrlinsolve(x.',C);
        w = temp*qrlinsolve(F',G);
    else
        % when matrix is fat, F is no longer square and we cannot play the
        % trick of thin matrix. Therefore, we have to form R2 and use LU.
        temp = qrlinsolve(x.',C);
        R2 = C'*temp;   % R2 = C'*R^(-1)*C
        [L2, U2] = lu(R2);
        temp2 = U2\(L2\G);
        w = temp*temp2;
    end
end
```







HDL Implementation Workflow

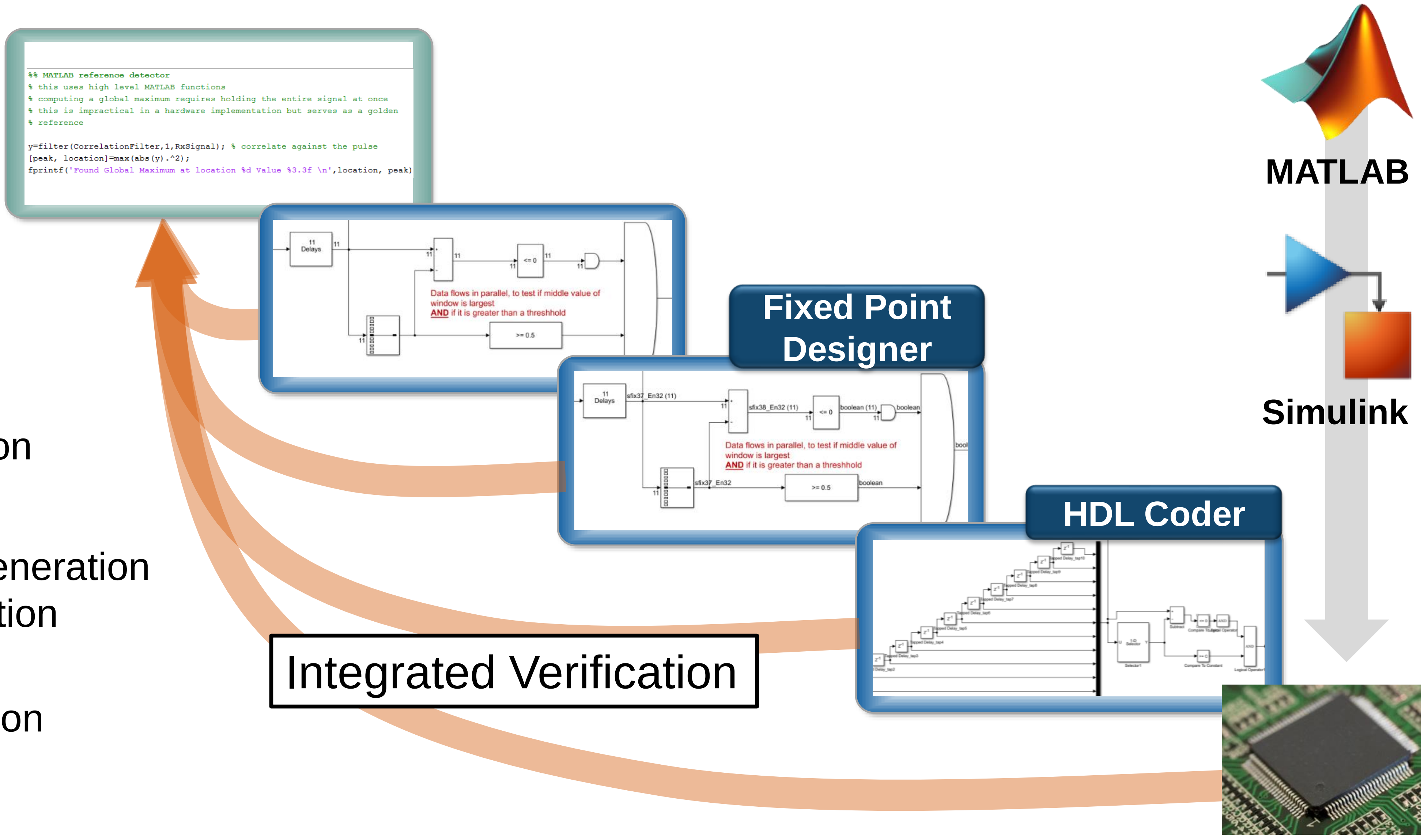
MATLAB Reference

Hardware Architecture

Fixed-point Implementation

HDL Code Generation and Optimization

HDL Verification and Targeting

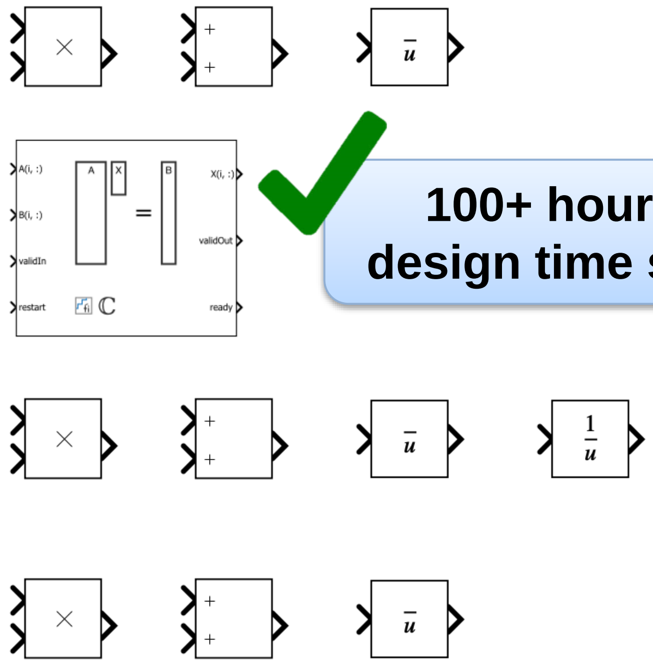


```
%% MATLAB reference detector
% this uses high level MATLAB functions
% computing a global maximum requires holding the entire signal at once
% this is impractical in a hardware implementation but serves as a golden
% reference

y=filter(CorrelationFilter,1,RxSignal); % correlate against the pulse
[peak, location]=max(abs(y).^2);
fprintf('Found Global Maximum at location %d Value %3.3f \n',location, peak)
```

Data flows in parallel, to test if middle value of window is largest
AND if it is greater than a threshhold

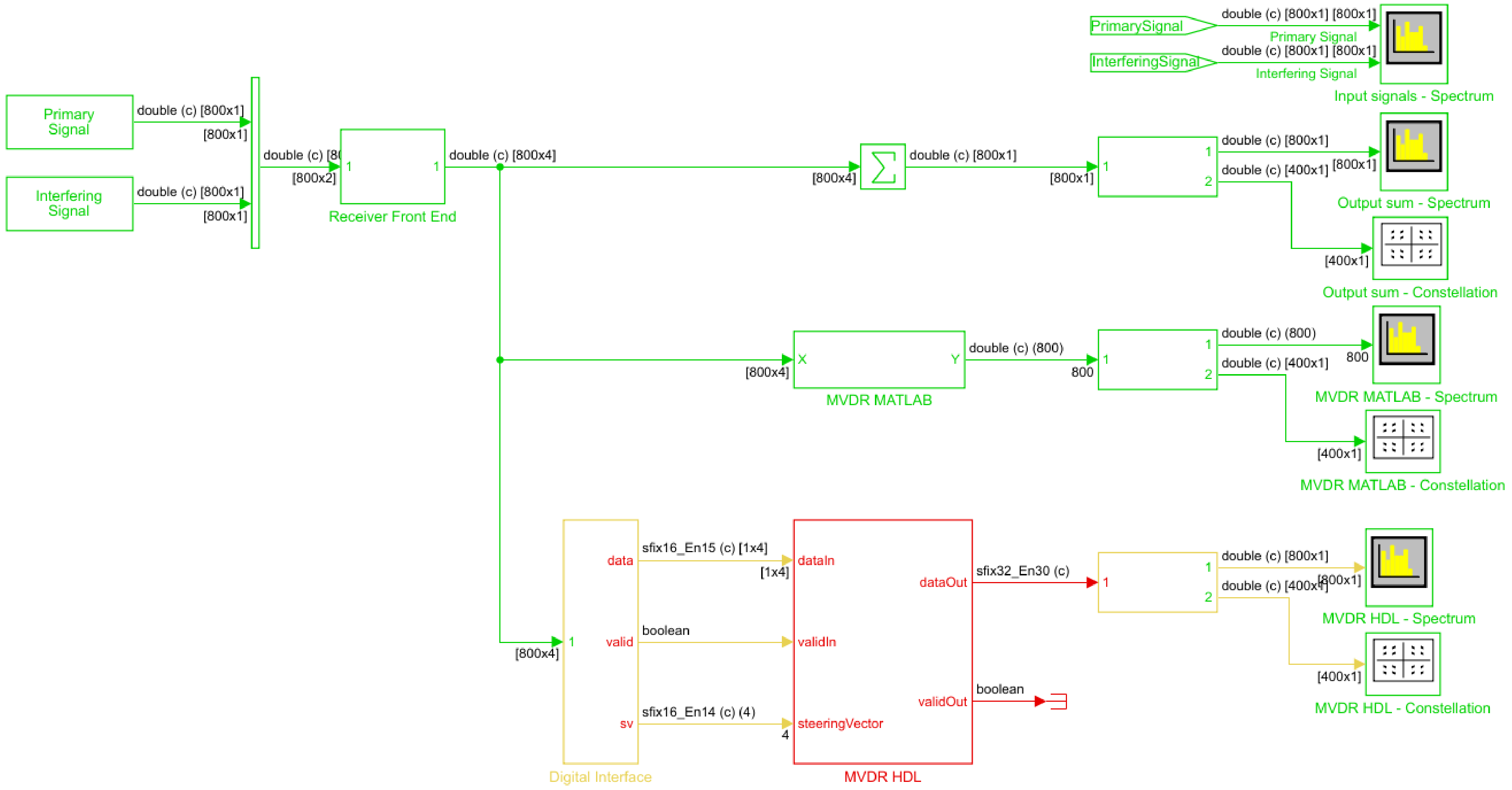
MATLAB MVDR reference code

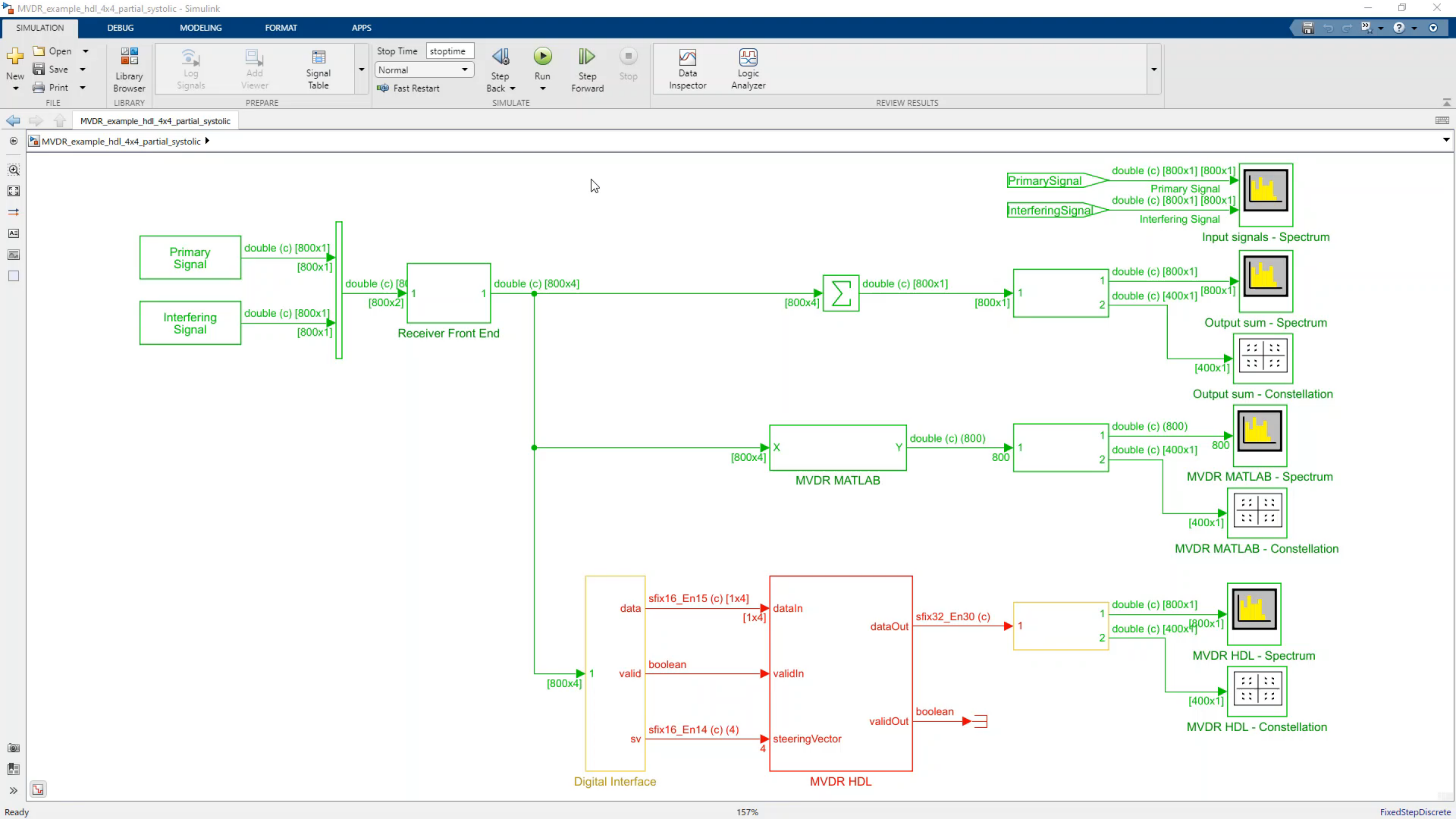
```
function Y = mvdr_beamform(X, sv)

% form covariance matrix
Ecx = X.'*conj(X);

% compute weight vector
wp = Ecx\sv;

% normalize response
w = wp/(sv'*wp);

% form output beam
Y = X*conj(w);

end
```



100+ hours of design time saved!

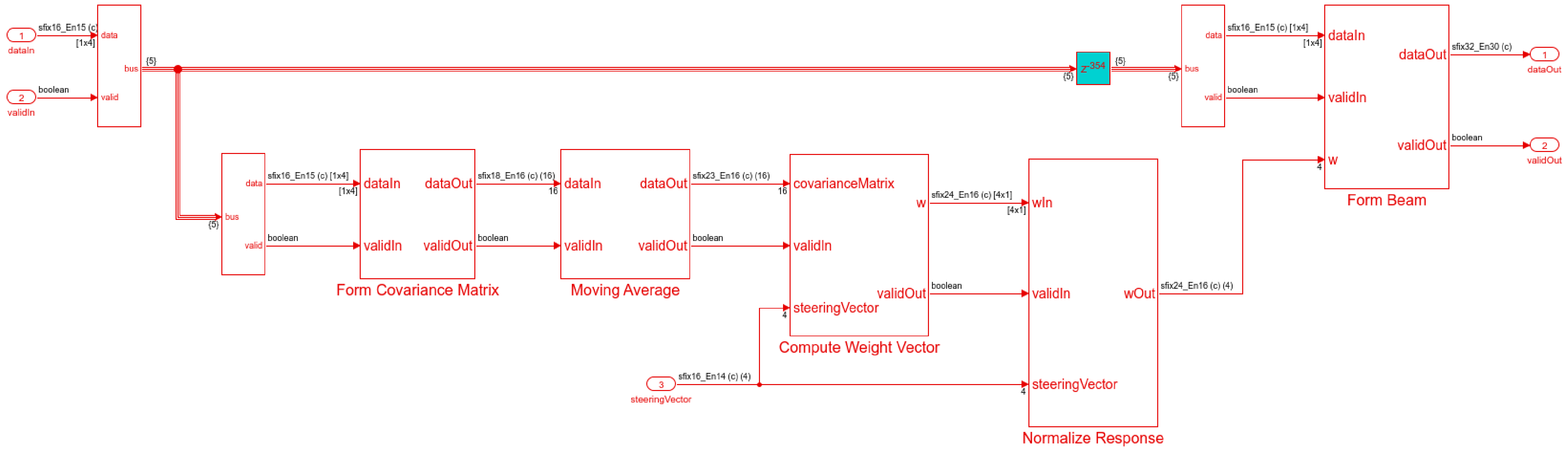
HDL Implementation of MVDR Beamforming



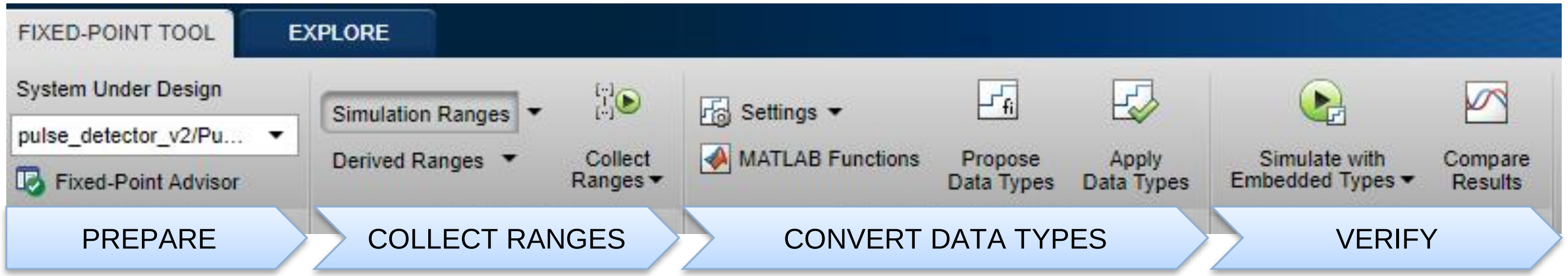

MVDR_example_hdl_4x4_partial_systolic - Simulink
SIMULATION
DEBUG
MODELING
FORMAT
APPS
New
Open
Save
Print
FILE
Library Browser
LIBRARY
Log Signals
Add Viewer
Signal Table
PREPARE
Stop Time
stoptime
Normal
Fast Restart
Step Back
Run
Step Forward
Stop
SIMULATE
Data Inspector
Logic Analyzer
REVIEW RESULTS
MVDR_example_hdl_4x4_partial_systolic
Primary Signal
double (c) [800x1]
[800x1]
Interfering Signal
double (c) [800x1]
[800x1]
double (c) [8
[800x2]
Receiver Front End
double (c) [800x4]
[800x4]
double (c) [800x1]
[800x1]
PrimarySignal
InterferingSignal
double (c) [800x1] [800x1]
Primary Signal
Interfering Signal
Input signals - Spectrum
double (c) [800x1]
double (c) [400x1]
Output sum - Spectrum
[400x1]
Output sum - Constellation
X
Y
MVDR MATLAB
double (c) (800)
800
double (c) [400x1]
MVDR MATLAB - Spectrum
MVDR MATLAB - Constellation
Digital Interface
data
valid
sv
sfix16_En15 (c) [1x4]
[1x4]
boolean
sfix16_En14 (c) (4)
4
dataIn
validIn
steeringVector
dataOut
validOut
sfix32_En30 (c)
MVDR HDL
MVDR HDL - Spectrum
MVDR HDL - Constellation
Ready
157%
FixedStepDiscrete

MATLAB EXPO

HDL Implementation of MVDR Beamforming



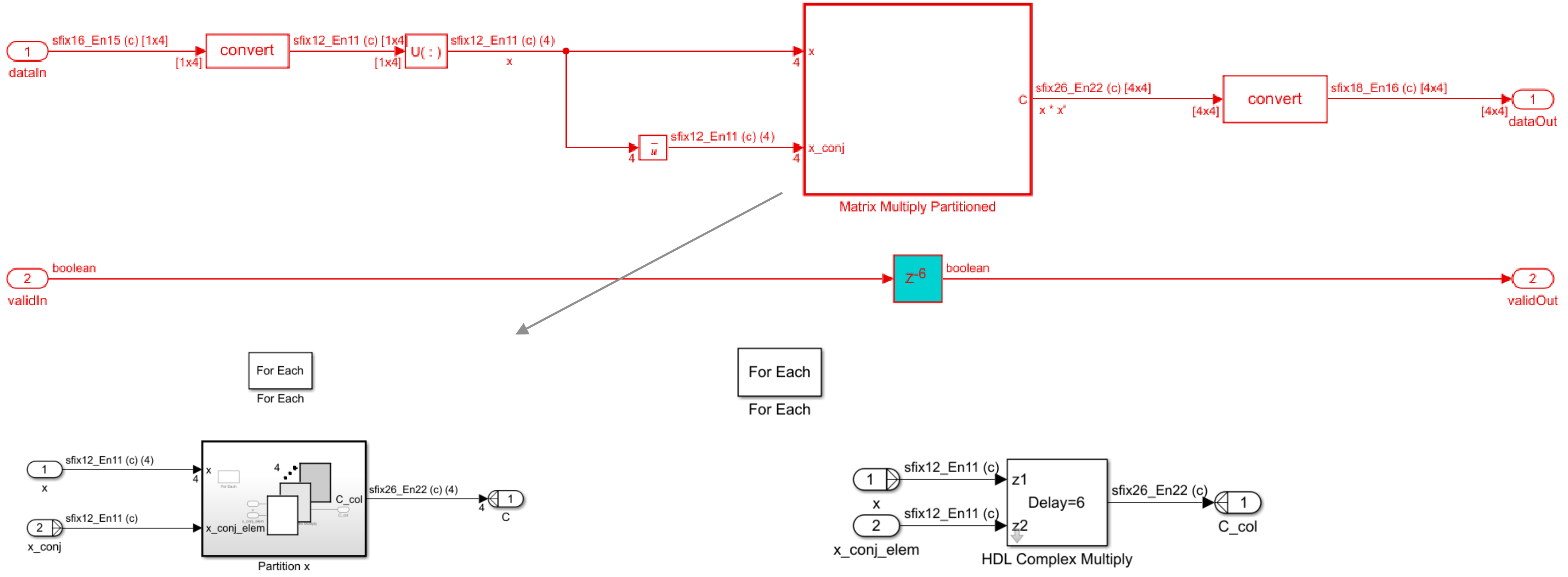
Fixed-Point Conversion using Fixed-Point Tool:



Form Covariance Matrix

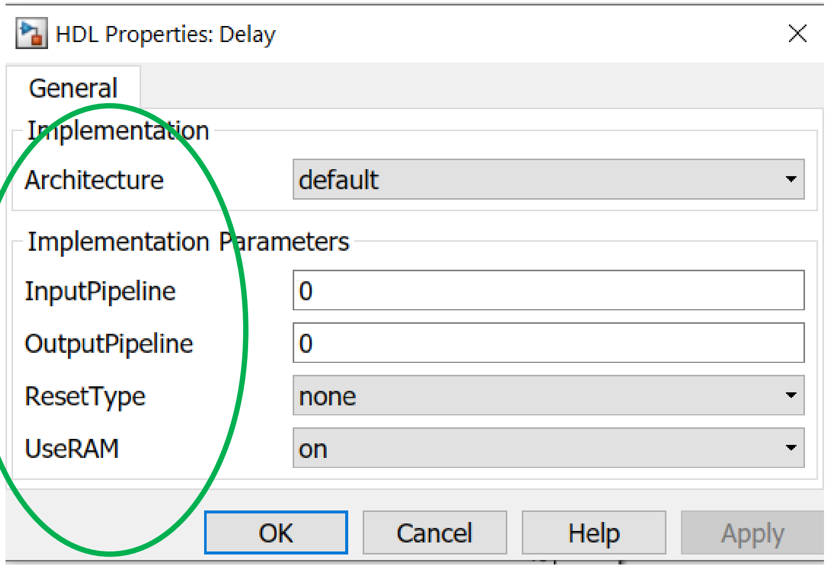
- For Each subsystem
  - Process elements independently
  - Concatenate results into outputs

```
% form covariance matrix
Ecx = X.'*conj(X);
```



Moving Average

- Use HDL Implementation properties to map large delays to Block RAM



HDL Properties: Delay

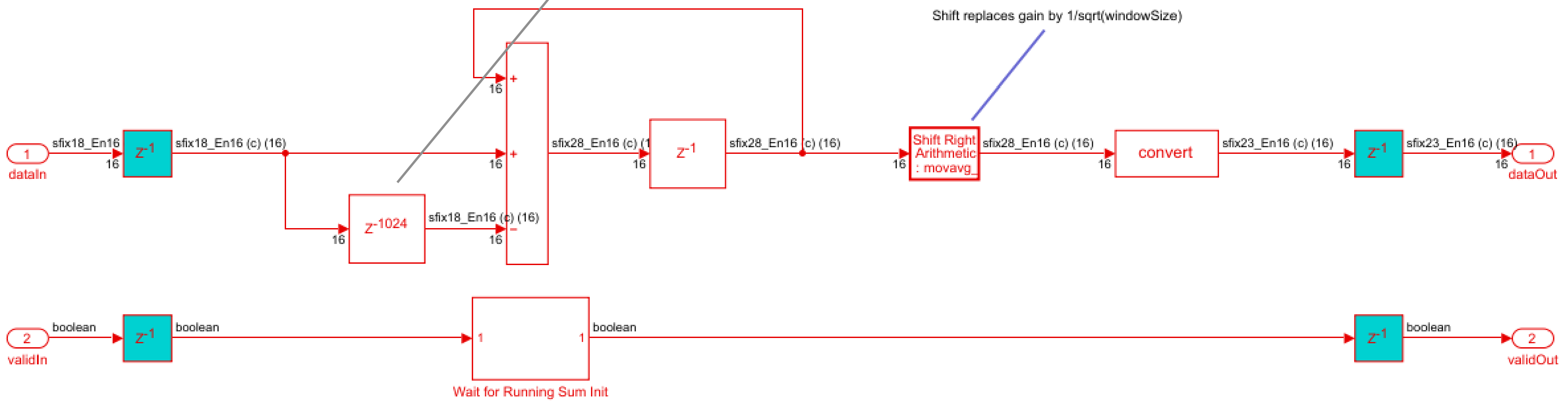
General

Implementation

Architecture	default

Implementation Parameters

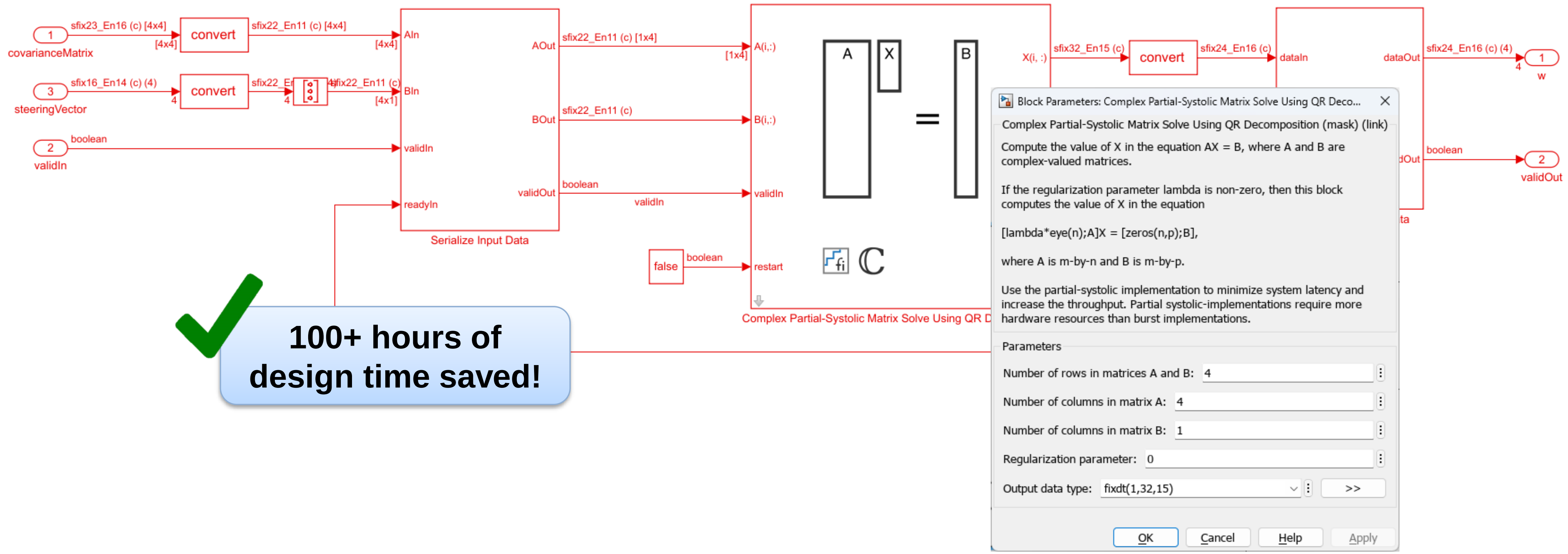
InputPipeline	0
OutputPipeline	0
ResetType	none
UseRAM	on



Compute Weight Vector

- Use Complex Matrix Solve block from Fixed-Point Matrix Linear Algebra Library

```
% compute weight vector
wp = Ecx\sv;
```



Block Parameters: Complex Partial-Systolic Matrix Solve Using QR Deco...

Complex Partial-Systolic Matrix Solve Using QR Decomposition (mask) (link)

Compute the value of X in the equation AX = B, where A and B are complex-valued matrices.

If the regularization parameter lambda is non-zero, then this block computes the value of X in the equation

[lambda*eye(n);A]X = [zeros(n,p);B],

where A is m-by-n and B is m-by-p.

Use the partial-systolic implementation to minimize system latency and increase the throughput. Partial systolic-implementations require more hardware resources than burst implementations.

Parameters

Number of rows in matrices A and B:	4
Number of columns in matrix A:	4
Number of columns in matrix B:	1
Regularization parameter:	0
Output data type:	fixdt(1,32,15)

OK Cancel Help Apply

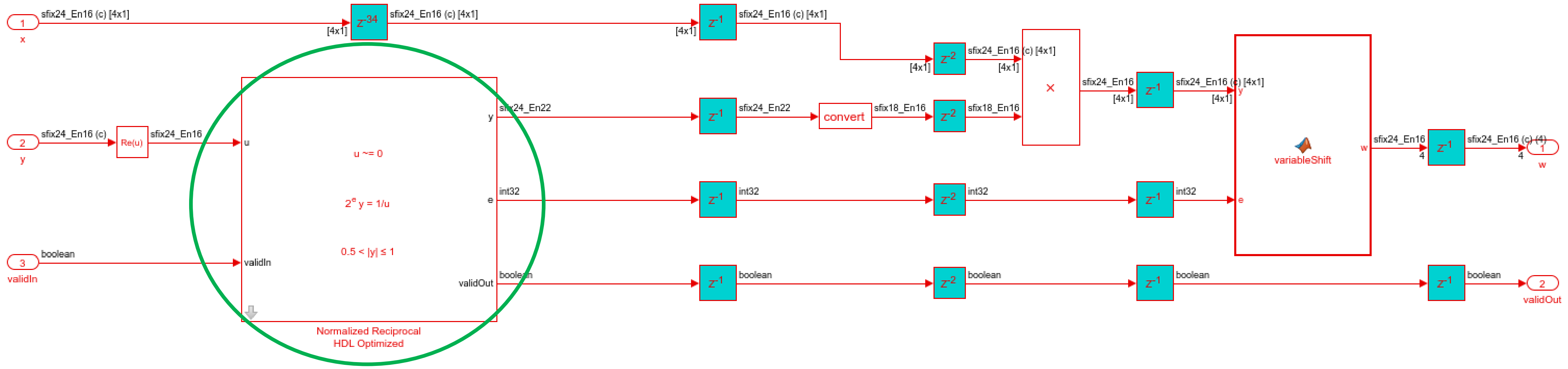
100+ hours of design time saved!

MATLAB EXPO

Normalize Response

- Perform divide using reciprocal and multiply
- Fixed-point CORDIC reciprocal “just works”

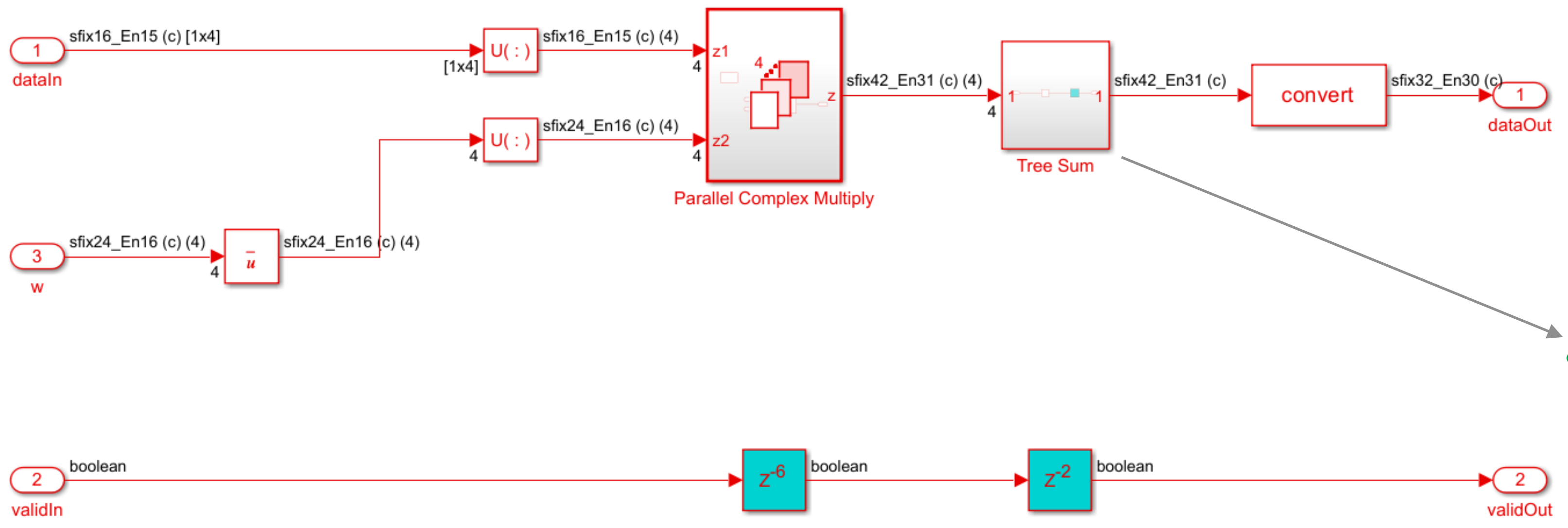
```
% normalize response
w = wp/(sv'*wp);
```

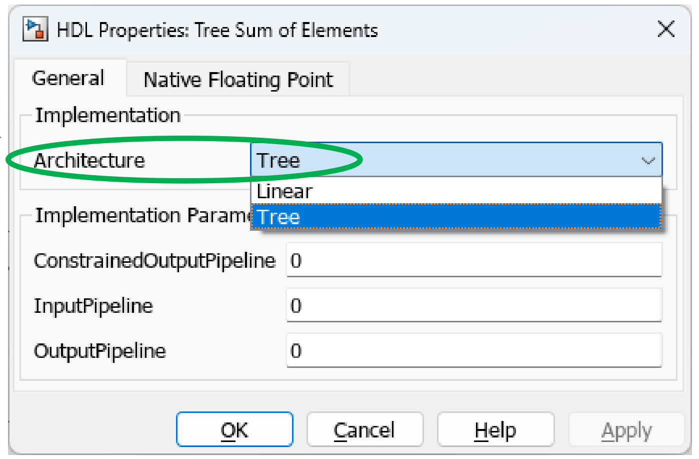


Form Output Beam

- Use HDL Block properties to set Architecture to Tree

```
% form output beam
Y = X*conj(w);
```




HDL Properties: Tree Sum of Elements

General | Native Floating Point

Implementation

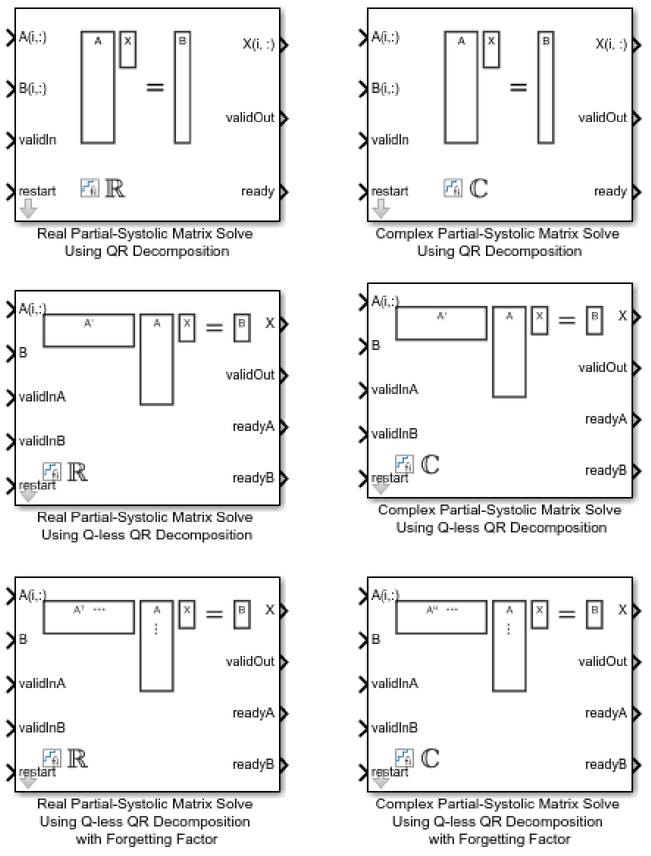
Architecture: Tree
- Linear
- Tree

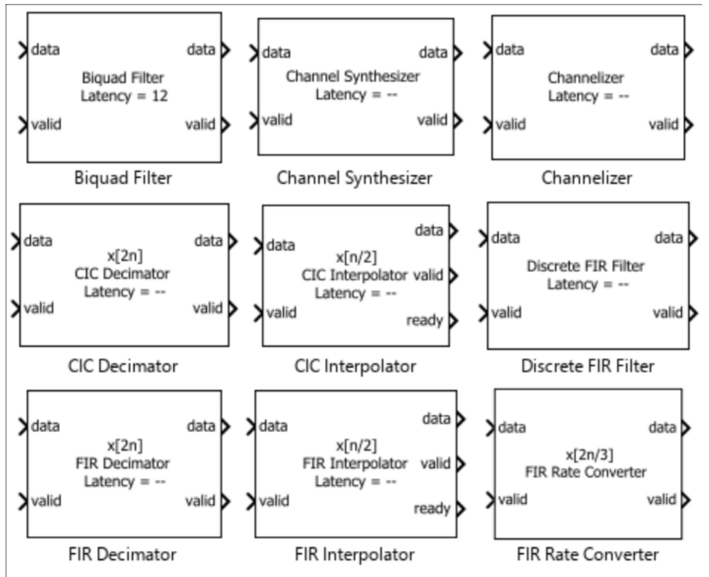
Implementation Parameters

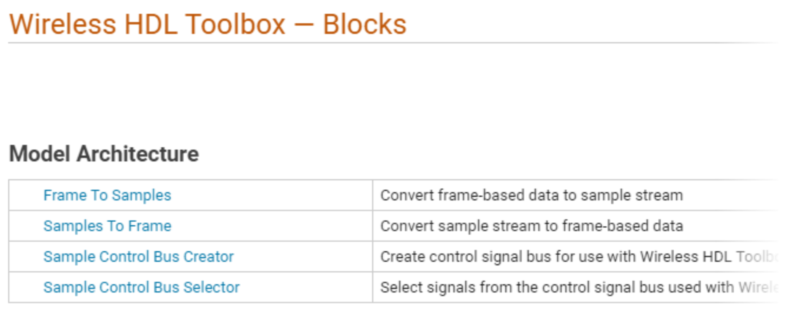
ConstrainedOutputPipeline	0
InputPipeline	0
OutputPipeline	0

Pre-verified, hardware-ready Simulink blocks and subsystems

Matrix solve, **GSPS** signal processing, Wireless, Vision & More..


Real Partial-Systolic Matrix Solve Using QR Decomposition
Complex Partial-Systolic Matrix Solve Using QR Decomposition
Real Partial-Systolic Matrix Solve Using Q-less QR Decomposition
Complex Partial-Systolic Matrix Solve Using Q-less QR Decomposition
Real Partial-Systolic Matrix Solve Using Q-less QR Decomposition with Forgetting Factor
Complex Partial-Systolic Matrix Solve Using Q-less QR Decomposition with Forgetting Factor
A(i,:)
B(i,:)
validIn
restart
X(i, :)
validOut
ready
validInA
validInB
readyA
readyB

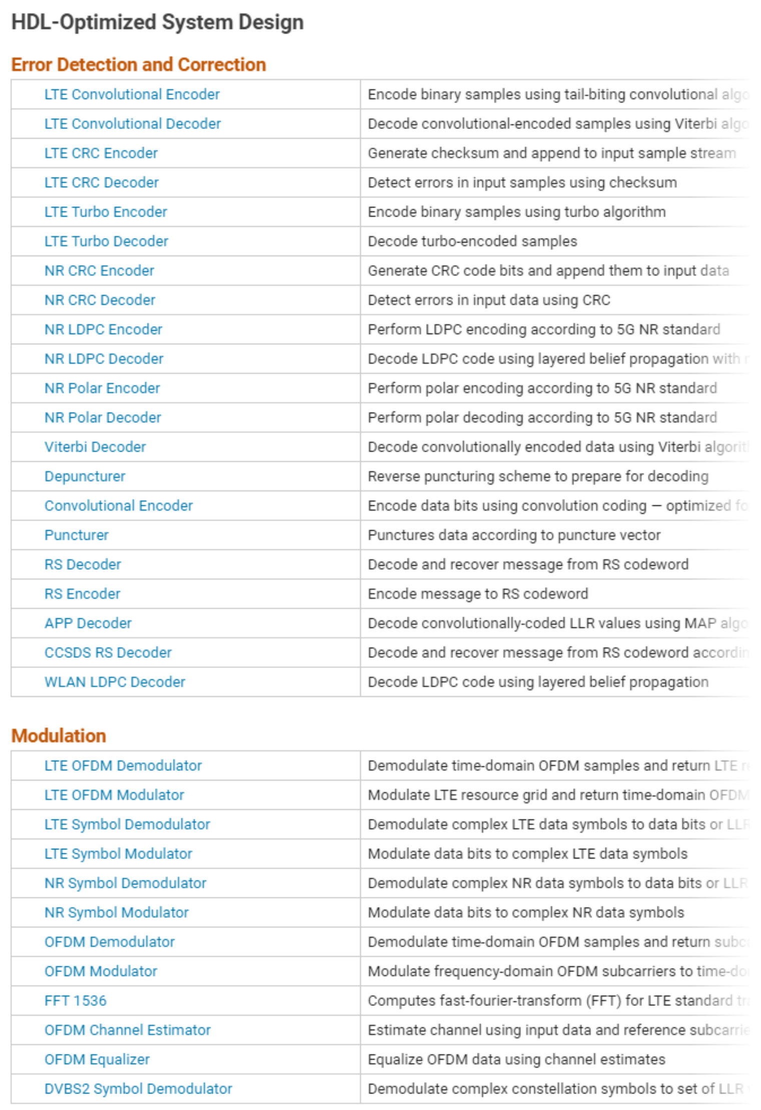




Wireless HDL Toolbox — Blocks

Model Architecture

Frame To Samples	Convert frame-based data to sample stream
Samples To Frame	Convert sample stream to frame-based data
Sample Control Bus Creator	Create control signal bus for use with Wireless HDL Toolb
Sample Control Bus Selector	Select signals from the control signal bus used with Wirele



HDL-Optimized System Design

Error Detection and Correction

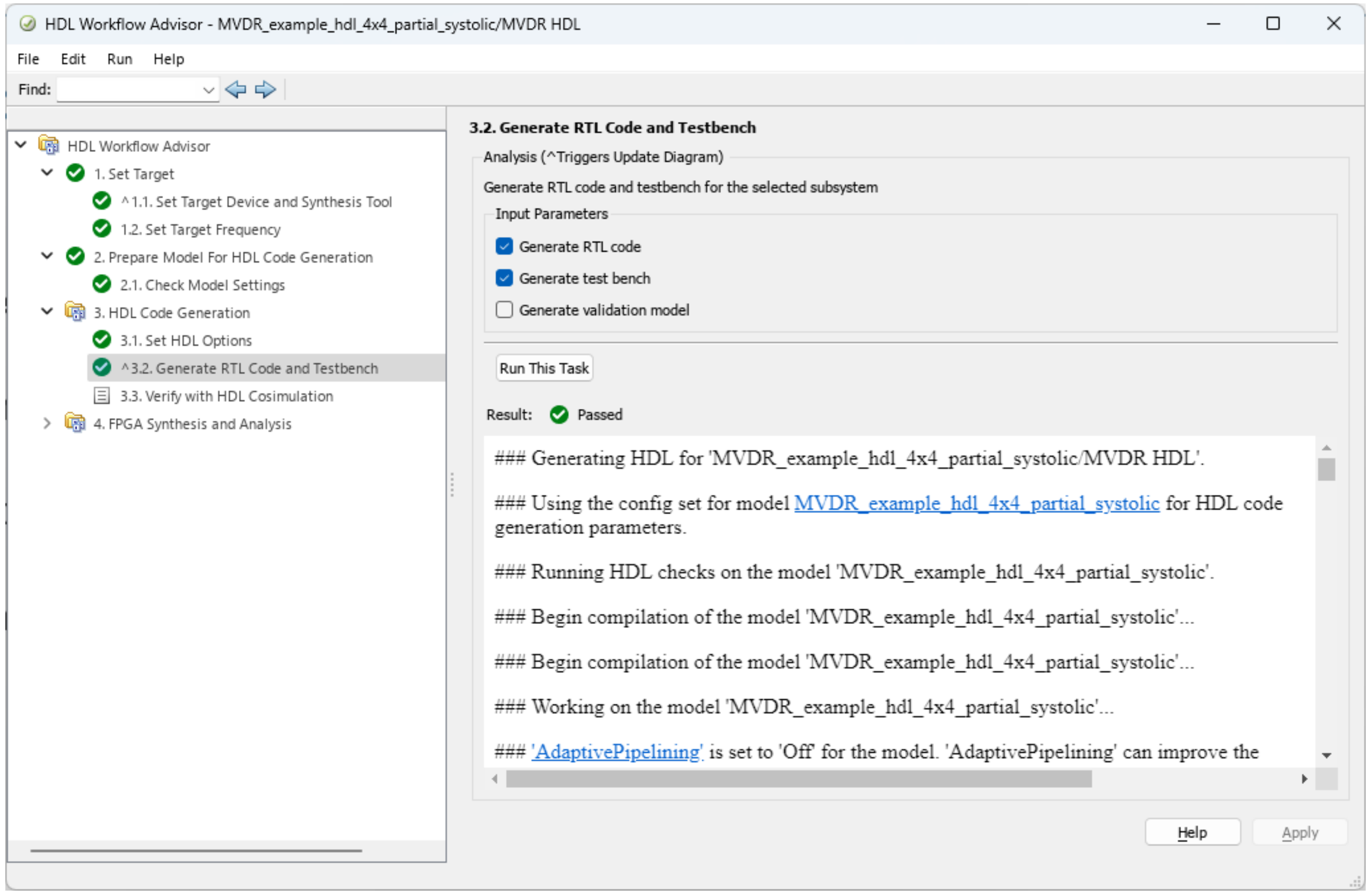
LTE Convolutional Encoder	Encode binary samples using tail-biting convolutional algo
LTE Convolutional Decoder	Decode convolutional-encoded samples using Viterbi algo
LTE CRC Encoder	Generate checksum and append to input sample stream
LTE CRC Decoder	Detect errors in input samples using checksum
LTE Turbo Encoder	Encode binary samples using turbo algorithm
LTE Turbo Decoder	Decode turbo-encoded samples
NR CRC Encoder	Generate CRC code bits and append them to input data
NR CRC Decoder	Detect errors in input data using CRC
NR LDPC Encoder	Perform LDPC encoding according to 5G NR standard
NR LDPC Decoder	Decode LDPC code using layered belief propagation with
NR Polar Encoder	Perform polar encoding according to 5G NR standard
NR Polar Decoder	Perform polar decoding according to 5G NR standard
Viterbi Decoder	Decode convolutionally encoded data using Viterbi algorit
Depuncturer	Reverse puncturing scheme to prepare for decoding
Convolutional Encoder	Encode data bits using convolution coding — optimized fo
Puncturer	Punctures data according to puncture vector
RS Decoder	Decode and recover message from RS codeword
RS Encoder	Encode message to RS codeword
APP Decoder	Decode convolutionally-coded LLR values using MAP algo
CCSDS RS Decoder	Decode and recover message from RS codeword accordin
WLAN LDPC Decoder	Decode LDPC code using layered belief propagation

Modulation

LTE OFDM Demodulator	Demodulate time-domain OFDM samples and return LTE r
LTE OFDM Modulator	Modulate LTE resource grid and return time-domain OFDM
LTE Symbol Demodulator	Demodulate complex LTE data symbols to data bits or LLR
LTE Symbol Modulator	Modulate data bits to complex LTE data symbols
NR Symbol Demodulator	Demodulate complex NR data symbols to data bits or LLR
NR Symbol Modulator	Modulate data bits to complex NR data symbols
OFDM Demodulator	Demodulate time-domain OFDM samples and return subc
OFDM Modulator	Modulate frequency-domain OFDM subcarriers to time-do
FFT 1536	Computes fast-fourier-transform (FFT) for LTE standard tr
OFDM Channel Estimator	Estimate channel using input data and reference subcarrie
OFDM Equalizer	Equalize OFDM data using channel estimates
DVBS2 Symbol Demodulator	Demodulate complex constellation symbols to set of LLR

Check, Generate and Synthesize HDL using Workflow Advisor

- Check model for HDL compatibility
- Generate HDL code and design summary
- Trace between HDL code and model
- Run synthesis and review results



HDL Workflow Advisor - MVDR_example_hdl_4x4_partial_systolic/MVDR HDL

- HDL Workflow Advisor
  - 1. Set Target
    - ^1.1. Set Target Device and Synthesis Tool
    - 1.2. Set Target Frequency
  - 2. Prepare Model For HDL Code Generation
    - 2.1. Check Model Settings
  - 3. HDL Code Generation
    - 3.1. Set HDL Options
    - ^3.2. Generate RTL Code and Testbench
    - 3.3. Verify with HDL Cosimulation
  - 4. FPGA Synthesis and Analysis

3.2. Generate RTL Code and Testbench

Analysis (^Triggers Update Diagram)

Generate RTL code and testbench for the selected subsystem

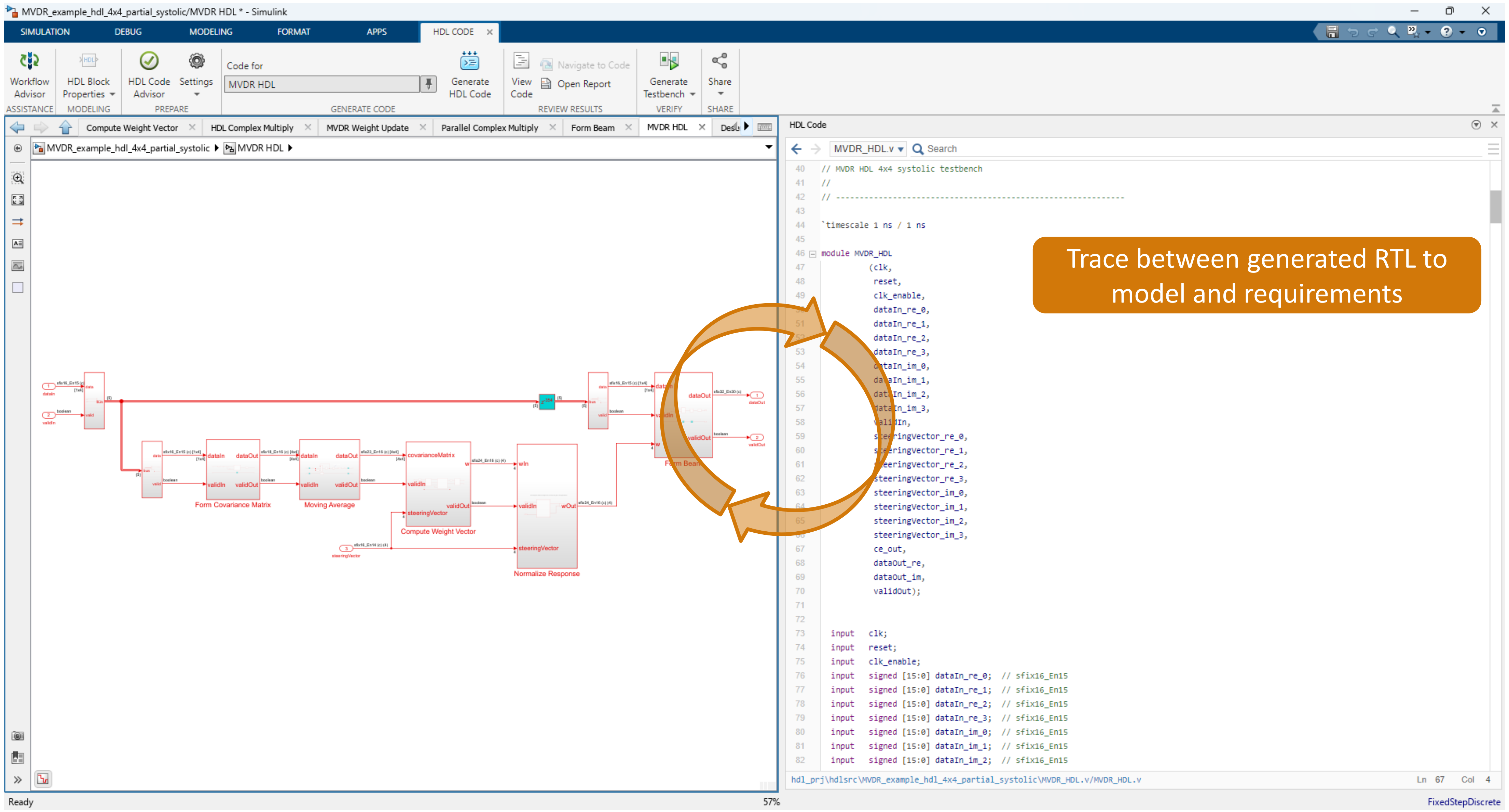
Input Parameters

- [✓] Generate RTL code
- [✓] Generate test bench
- [ ] Generate validation model

Result: ✓ Passed

```
### Generating HDL for 'MVDR_example_hdl_4x4_partial_systolic/MVDR HDL'.
### Using the config set for model MVDR_example_hdl_4x4_partial_systolic for HDL code generation parameters.
### Running HDL checks on the model 'MVDR_example_hdl_4x4_partial_systolic'.
### Begin compilation of the model 'MVDR_example_hdl_4x4_partial_systolic'...
### Begin compilation of the model 'MVDR_example_hdl_4x4_partial_systolic'...
### Working on the model 'MVDR_example_hdl_4x4_partial_systolic'...
### 'AdaptivePipelining' is set to 'Off' for the model. 'AdaptivePipelining' can improve the
```

Readable, Traceable RTL Code

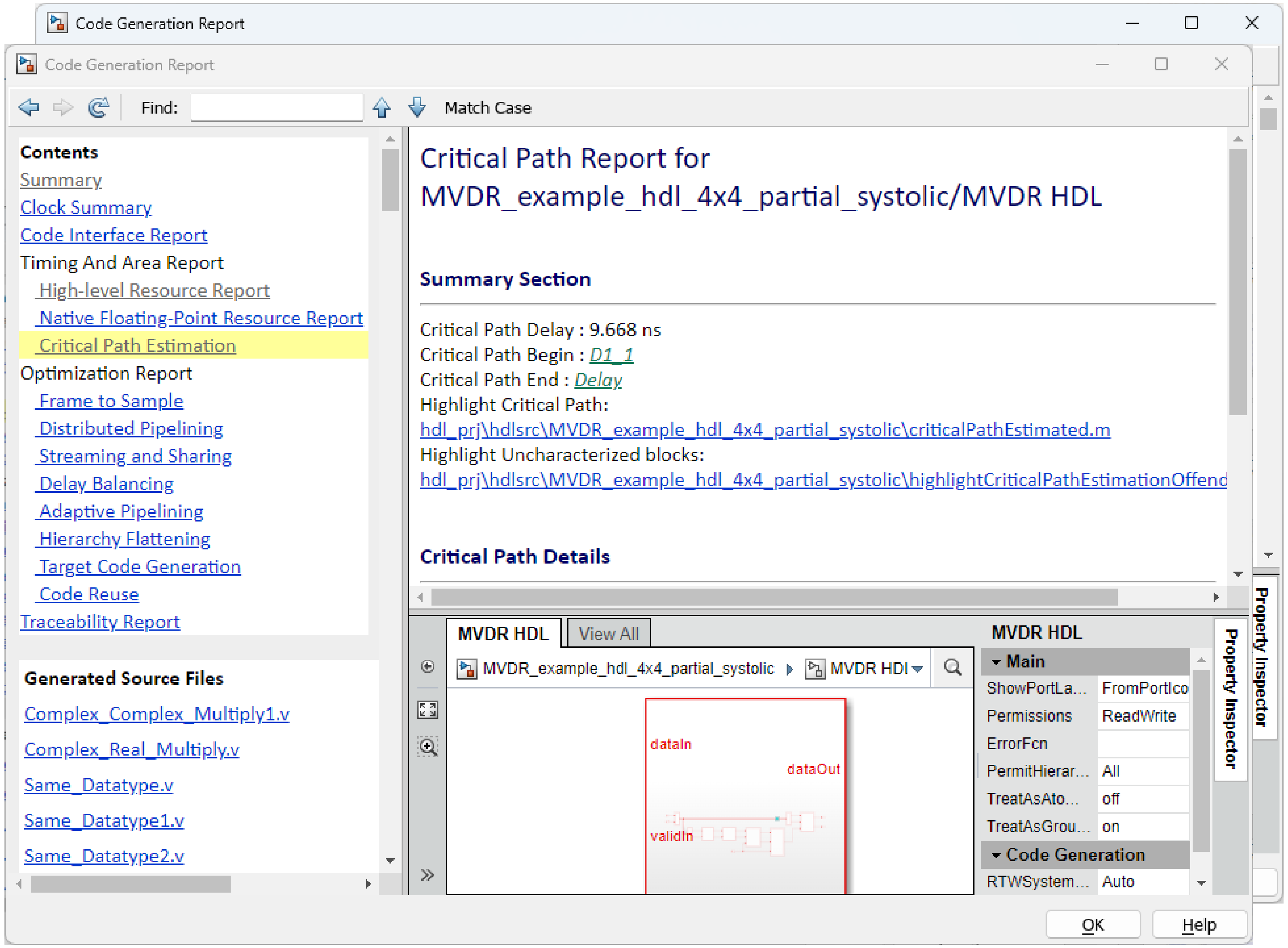


```
// MVDR HDL 4x4 systolic testbench
//
// -------------------------------------------------------------

`timescale 1 ns / 1 ns

module MVDR_HDL
          (clk,
           reset,
           clk_enable,
           dataIn_re_0,
           dataIn_re_1,
           dataIn_re_2,
           dataIn_re_3,
           dataIn_im_0,
           dataIn_im_1,
           dataIn_im_2,
           dataIn_im_3,
           validIn,
           steeringVector_re_0,
           steeringVector_re_1,
           steeringVector_re_2,
           steeringVector_re_3,
           steeringVector_im_0,
           steeringVector_im_1,
           steeringVector_im_2,
           steeringVector_im_3,
           ce_out,
           dataOut_re,
           dataOut_im,
           validOut);


  input   clk;
  input   reset;
  input   clk_enable;
  input   signed [15:0] dataIn_re_0;  // sfix16_En15
  input   signed [15:0] dataIn_re_1;  // sfix16_En15
  input   signed [15:0] dataIn_re_2;  // sfix16_En15
  input   signed [15:0] dataIn_re_3;  // sfix16_En15
  input   signed [15:0] dataIn_im_0;  // sfix16_En15
  input   signed [15:0] dataIn_im_1;  // sfix16_En15
  input   signed [15:0] dataIn_im_2;  // sfix16_En15
```

Trace between generated RTL to model and requirements

Code Generation Report

- Resource Utilization Report
- Critical Path Estimation Report



Code Generation Report

Find: Match Case

Contents

Summary
Clock Summary
Code Interface Report
Timing And Area Report
High-level Resource Report
Native Floating-Point Resource Report
Critical Path Estimation
Optimization Report
Frame to Sample
Distributed Pipelining
Streaming and Sharing
Delay Balancing
Adaptive Pipelining
Hierarchy Flattening
Target Code Generation
Code Reuse
Traceability Report

Generated Source Files

Complex_Complex_Multiply1.v
Complex_Real_Multiply.v
Same_Datatype.v
Same_Datatype1.v
Same_Datatype2.v

Critical Path Report for
MVDR_example_hdl_4x4_partial_systolic/MVDR HDL

Summary Section

Critical Path Delay : 9.668 ns
Critical Path Begin : D1_1
Critical Path End : Delay
Highlight Critical Path:
hdl_prj\hdlsrc\MVDR_example_hdl_4x4_partial_systolic\criticalPathEstimated.m
Highlight Uncharacterized blocks:
hdl_prj\hdlsrc\MVDR_example_hdl_4x4_partial_systolic\highlightCriticalPathEstimationOffend

Critical Path Details

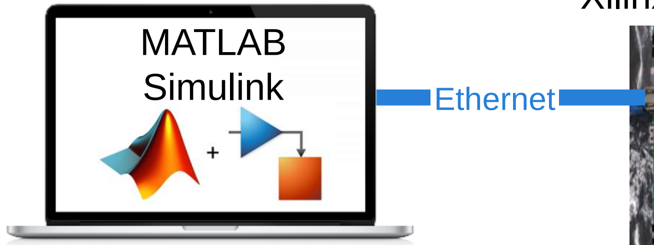
MVDR HDL

Main	
ShowPortLa...	FromPortIco
Permissions	ReadWrite
ErrorFcn	
PermitHierar...	All
TreatAsAto...	off
TreatAsGrou...	on
Code Generation	
RTWSystem...	Auto

Implementation Results

- Device: xczu49dr (ZCU216)
- Maximum frequency: 370 MHz
- Resource utilization:

Resource	Utilization	(%)
LUT	46K	10.9
LUTRAM	1.5K	0.7
FF	37K	4.3
BRAM	18	1.7
DSP	94	2.2



Xilinx ZCU216 RFSoC Eval Board

Leaving you still a lot of room for integration!

RF Pixels Verifies Millimeter Wave RF Electronics on a Zynq RFSoC Based Digital Baseband

Challenge

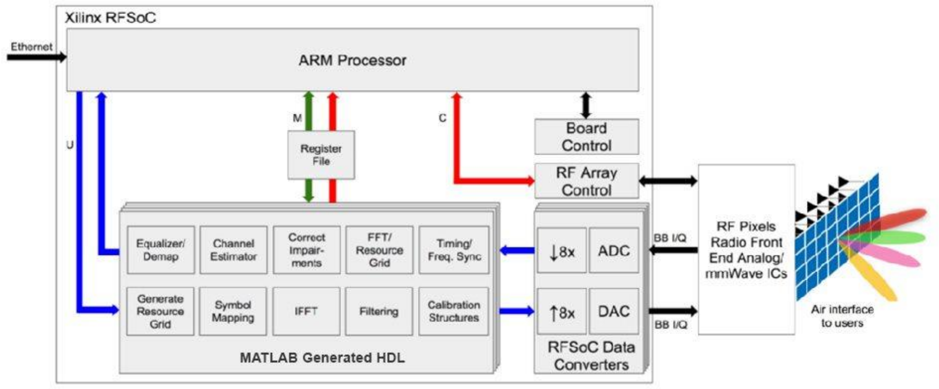
Test and demonstrate radio front-end designs that incorporate specialized RF electronics hardware and millimeter wave spectrum technology

Solution

Use MATLAB and Simulink to implement a digital baseband and deploy it to a Zynq RFSoC board for over-the-air testing

Results

- Engineering effort reduced by one year or more
- Digital baseband implementation completed by a single engineer
- Design iterations reduced from weeks to days



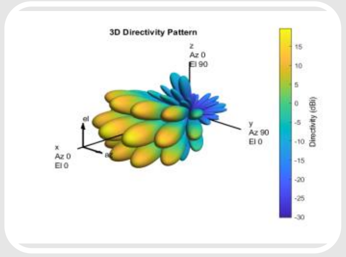
Digital baseband implemented in HDL, used to verify the RF Pixels radio front end.

"By adapting the LTE golden reference model from Wireless HDL Toolbox and deploying it to a Zynq UltraScale+ RFSoC board using HDL Coder, we saved us at least a year of engineering effort—and this approach enabled me to complete the implementation myself, without having to hire an additional digital engineer."

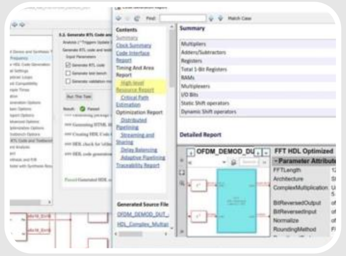
- Matthew Weiner, RF Pixels

Link to technical article

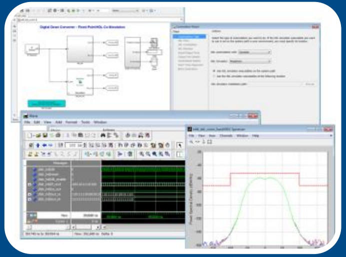
Key Takeaways



Beamforming for interference mitigation



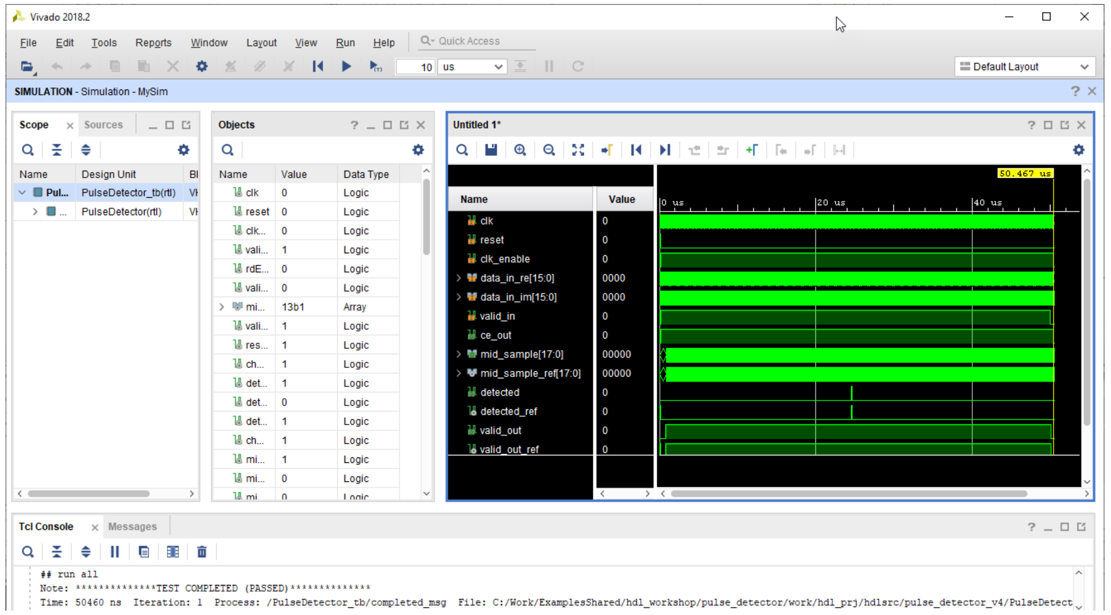
HDL Code Generation for Beamforming algorithms



Integrated Verification of HDL Code

Automated HDL Test Bench

- Generate HDL test bench with test vectors captured from Simulink model
- Run HDL simulation (using Vivado Simulator, Questa, etc) to verify correctness of generated RTL

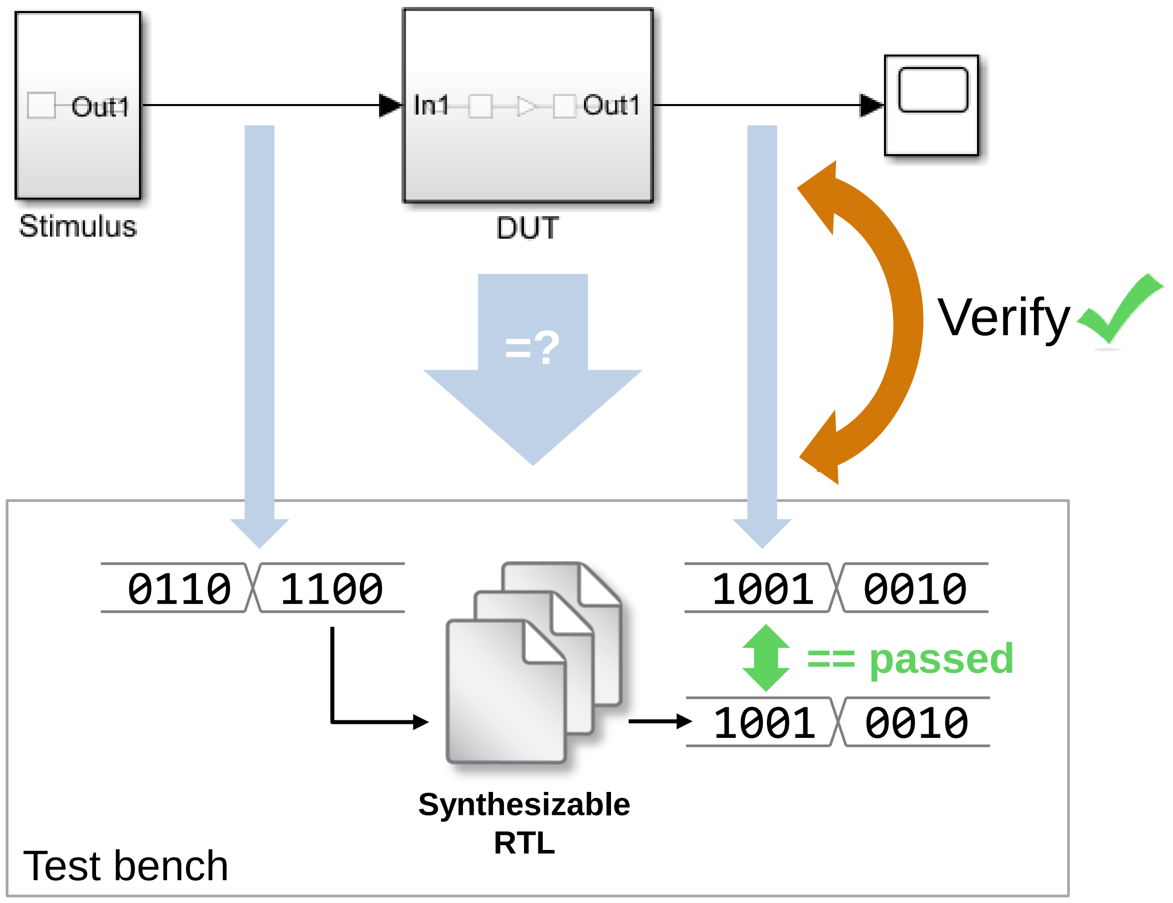


Name	Design Unit	Bl
Pul...	PulseDetector_tb(rtl)	VI
...	PulseDetector(rtl)	VI

Name	Value	Data Type
clk	0	Logic
reset	0	Logic
clk...	0	Logic
vali...	1	Logic
rdE...	0	Logic
vali...	0	Logic
mi...	13b1	Array
vali...	1	Logic
res...	1	Logic
ch...	1	Logic
det...	1	Logic
det...	0	Logic
det...	1	Logic
ch...	1	Logic
mi...	1	Logic
mi...	0	Logic
mi	0	Logic

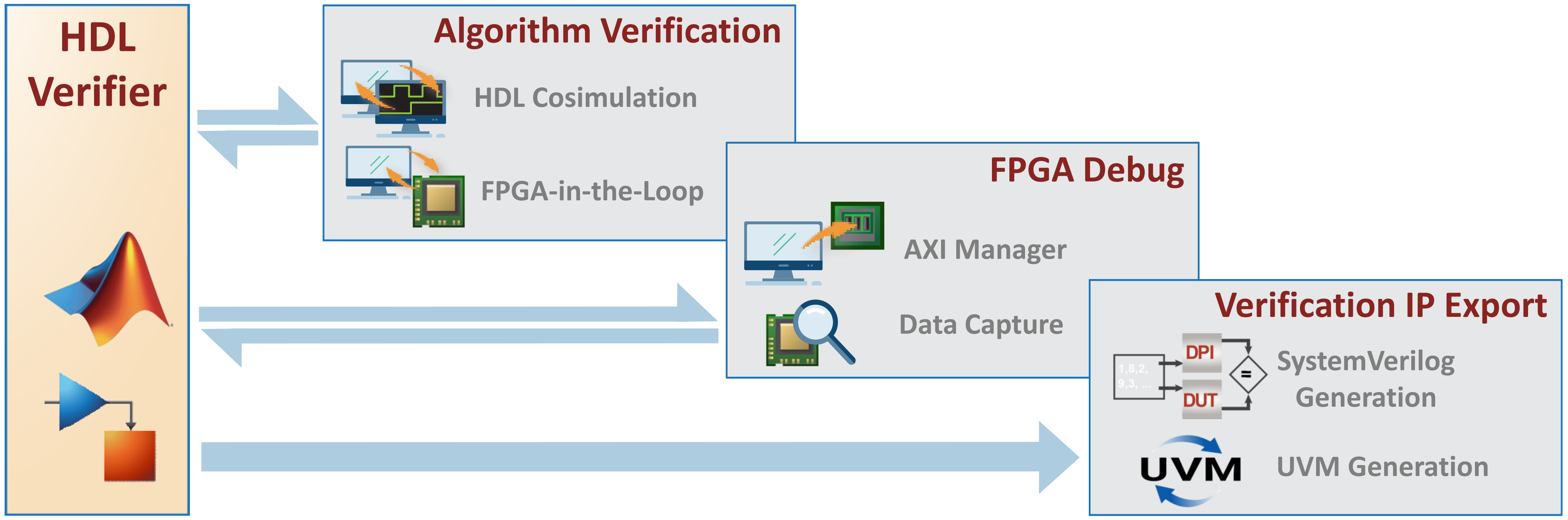
Name	Value
clk	0
reset	0
clk_enable	0
data_in_re[15:0]	0000
data_in_im[15:0]	0000
valid_in	0
ce_out	0
mid_sample[17:0]	00000
mid_sample_ref[17:0]	00000
detected	0
detected_ref	0
valid_out	0
valid_out_ref	0

```
## run all
Note: **************TEST COMPLETED (PASSED)**************
Time: 50460 ns  Iteration: 1  Process: /PulseDetector_tb/completed_msg  File: C:/Work/ExamplesShared/hdl_workshop/pulse_detector/work/hdl_prj/hdlsrc/pulse_detector_v4/PulseDetect
```



HDL Verifier

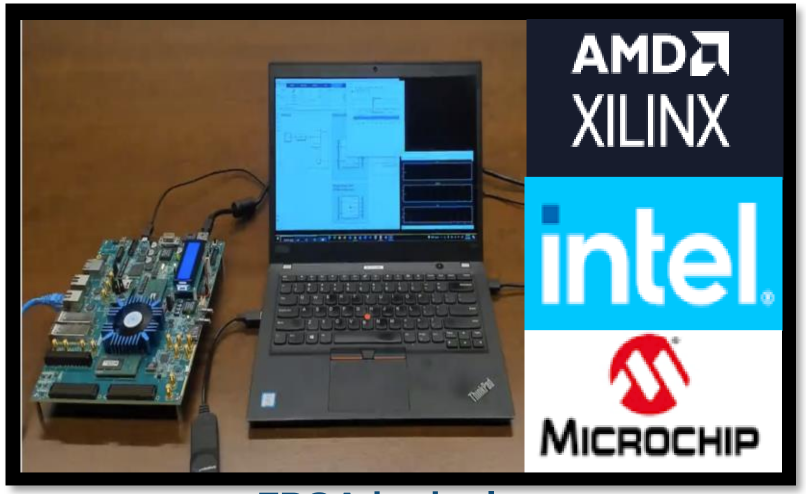
Test and verify Verilog and VHDL using HDL simulators and FPGA boards



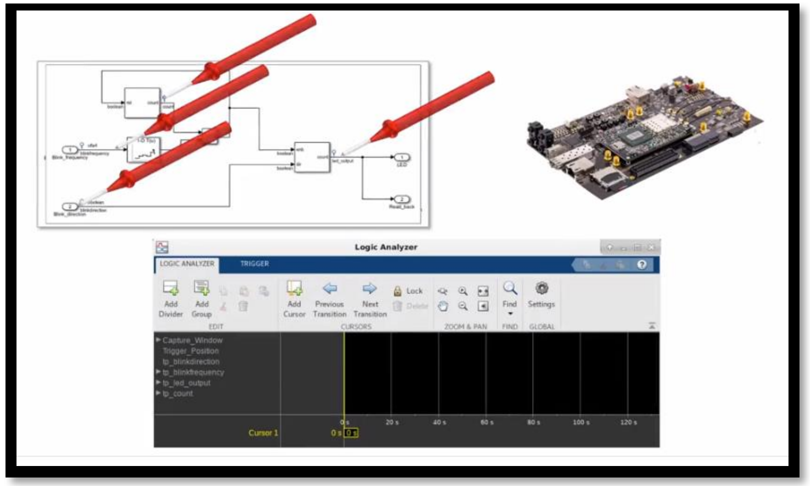
HDL Verifier

Test and verify Verilog and VHDL using HDL simulators and FPGA boards

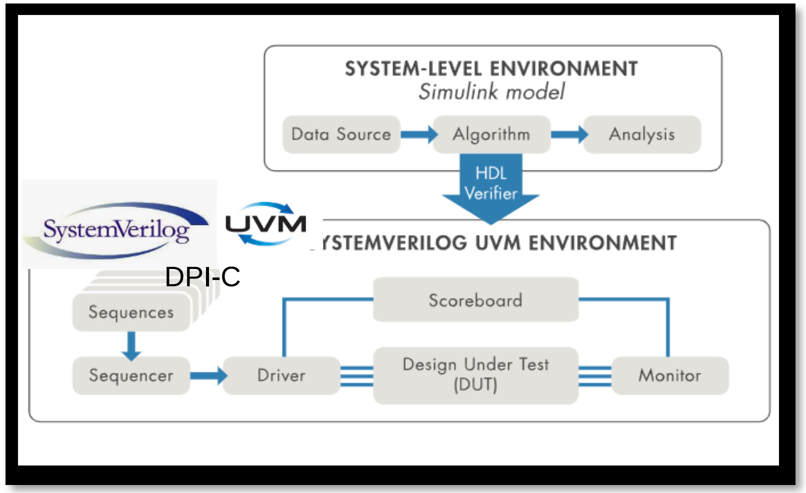
HDL Co-simulation



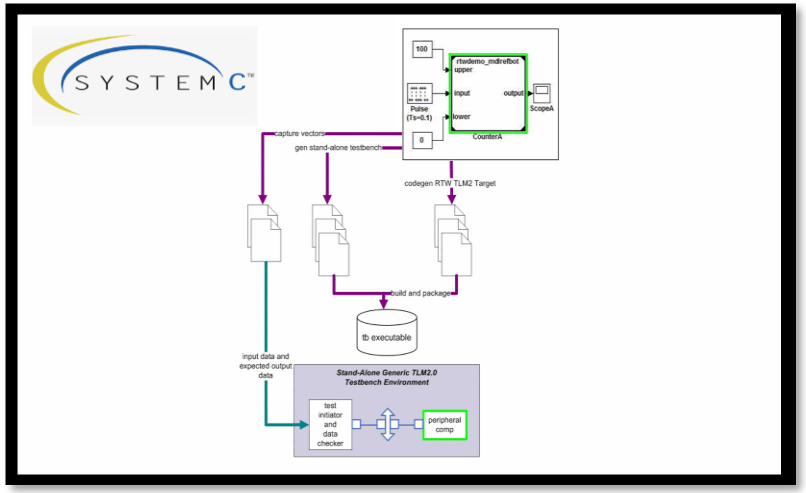
FPGA-in-the-loop



FPGA Debugging (Data Capture and AXI Manager)

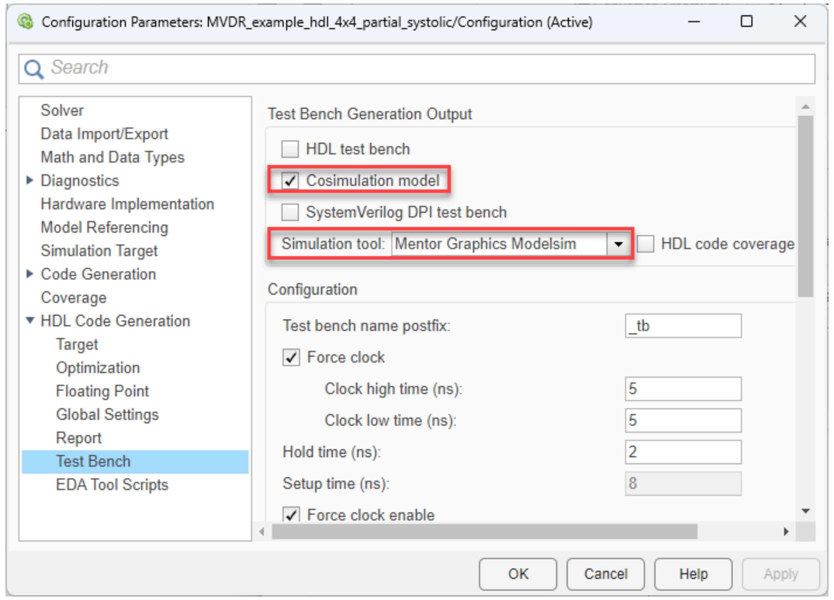


UVM Testbench / SystemVerilog DPI-C Test Components Generation


SYSTEM C
capture vectors
gen stand-alone testbench
codegen RTW TLM2 Target
build and package
tb executable
input data and expected output data
Stand-Alone Generic TLM2.0 Testbench Environment
test initiator and data checker
peripheral comp

System-C TLM 2.0 Components Generation

HDL Cosimulation Setup Creation



Configuration Parameters: MVDR_example_hdl_4x4_partial_systolic/Configuration (Active)

Solver
Data Import/Export
Math and Data Types
Diagnostics
Hardware Implementation
Model Referencing
Simulation Target
Code Generation
Coverage
HDL Code Generation
Target
Optimization
Floating Point
Global Settings
Report
Test Bench
EDA Tool Scripts

Test Bench Generation Output

- [ ] HDL test bench
- [x] Cosimulation model
- [ ] SystemVerilog DPI test bench

Simulation tool: Mentor Graphics Modelsim

- [ ] HDL code coverage

Configuration

Test bench name postfix: _tb

- [x] Force clock

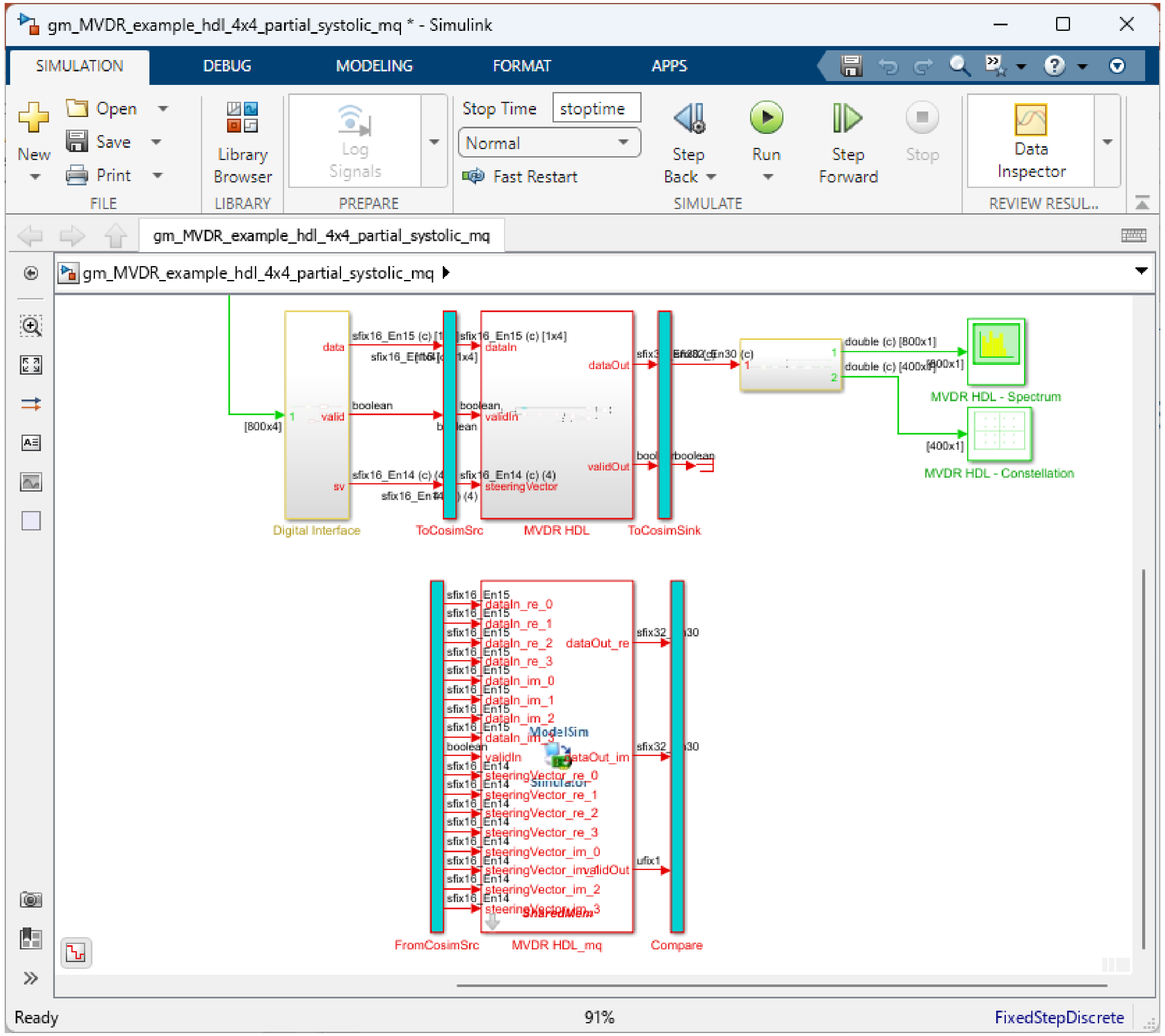
Clock high time (ns): 5

Clock low time (ns): 5

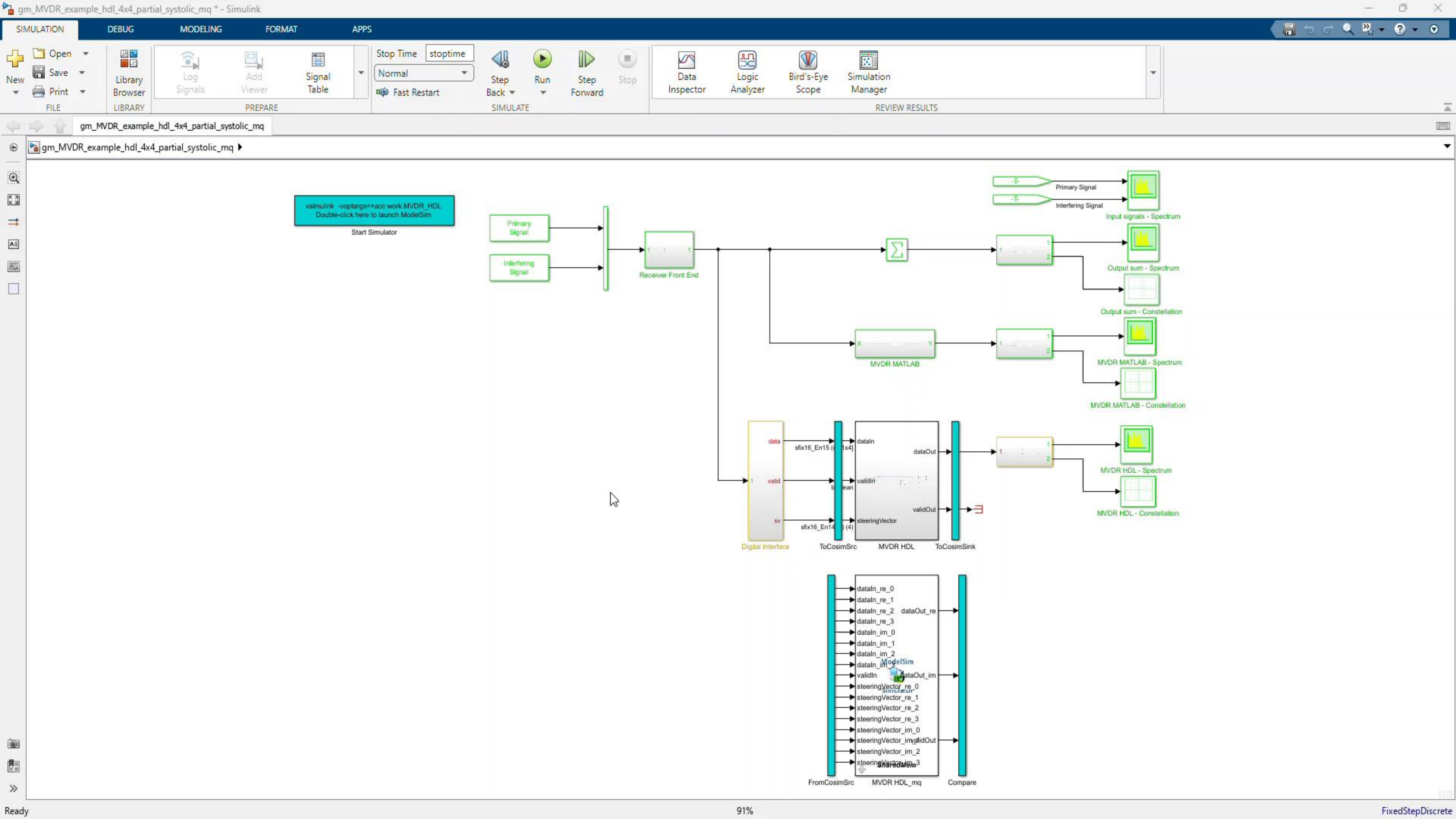
Hold time (ns): 2

Setup time (ns): 8

- [x] Force clock enable



gm_MVDR_example_hdl_4x4_partial_systolic_mq * - Simulink



vsimulink -voptargs=+acc work.MVDR_HDL

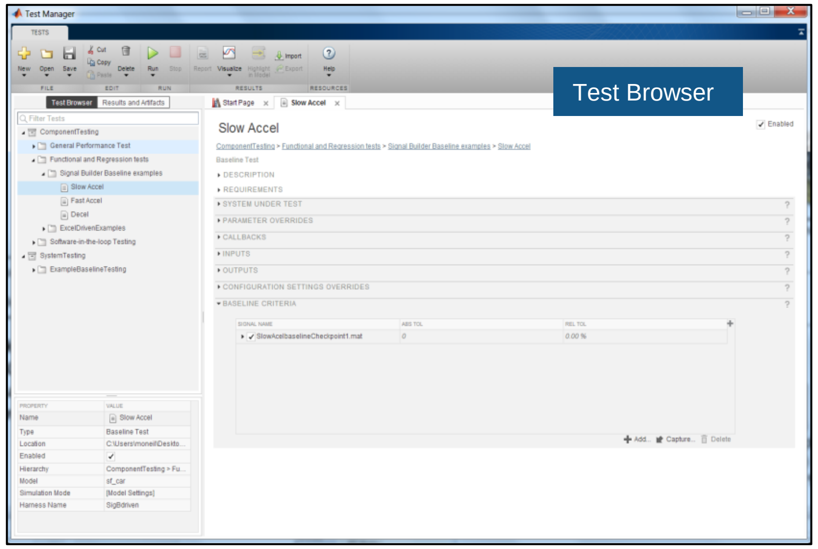
Combining with power of Simulink Test Manager

Develop, manage, and execute simulation-based tests

Author, manage, organize tests

Execute simulation, equivalence and baseline tests

Review, export and report



ComponentTesting
- General Performance Test
- Functional and Regression tests
  - Signal Builder Baseline examples
    - Slow Accel
    - Fast Accel
    - Decel
  - ExcelDrivenExamples
- Software-in-the-loop Testing

SystemTesting
- ExampleBaselineTesting

Slow Accel

ComponentTesting > Functional and Regression tests > Signal Builder Baseline examples > Slow Accel

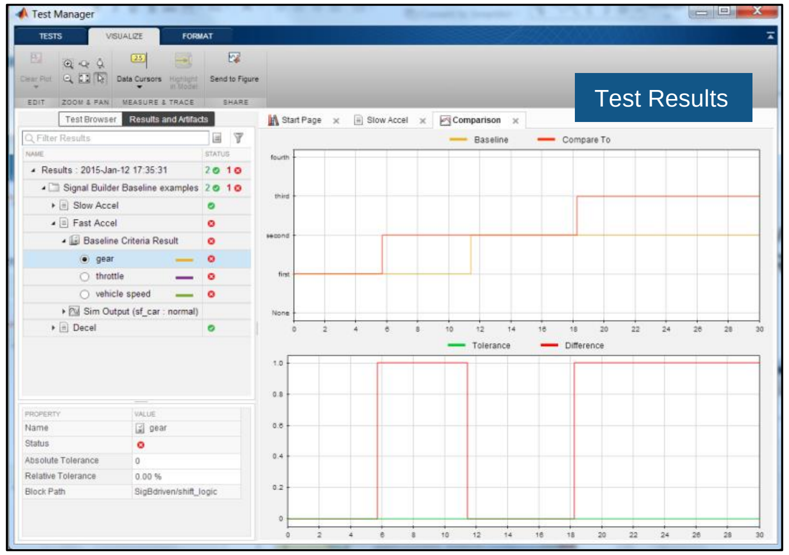
Baseline Test

DESCRIPTION
REQUIREMENTS
SYSTEM UNDER TEST
PARAMETER OVERRIDES
CALLBACKS
INPUTS
OUTPUTS
CONFIGURATION SETTINGS OVERRIDES
BASELINE CRITERIA

SIGNAL NAME	ABS TOL	REL TOL
✓ SlowAccelbaselineCheckpoint1.mat	0	0.00 %

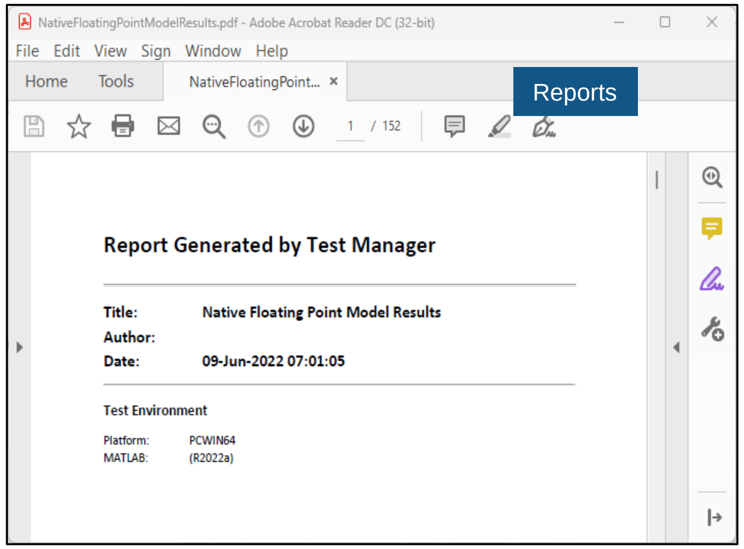
Add... Capture... Delete

PROPERTY	VALUE
Name	Slow Accel
Type	Baseline Test
Location	C:\Users\moneil\Deskto...
Enabled	✓
Hierarchy	ComponentTesting > Fu...
Model	sf_car
Simulation Mode	[Model Settings]
Harness Name	SigBdriven



NAME	STATUS
Results : 2015-Jan-12 17:35:31	2 ✓ 1 ✗
Signal Builder Baseline examples	2 ✓ 1 ✗
Slow Accel	✓
Fast Accel	✗
Baseline Criteria Result	✗
gear	✗
throttle	✗
vehicle speed	✗
Sim Output (sf_car : normal)	
Decel	✓

PROPERTY	VALUE
Name	gear
Status	✗
Absolute Tolerance	0
Relative Tolerance	0.00 %
Block Path	SigBdriven/shift_logic



Report Generated by Test Manager

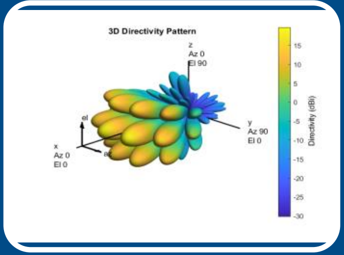
Title:	Native Floating Point Model Results
Author:	
Date:	09-Jun-2022 07:01:05

Test Environment

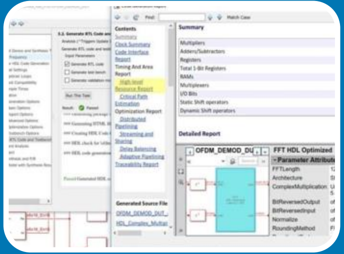
Platform:	PCWIN64
MATLAB:	(R2022a)

Summary and Resources

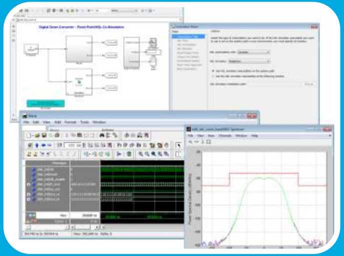
Key Takeaways



Beamforming for interference mitigation

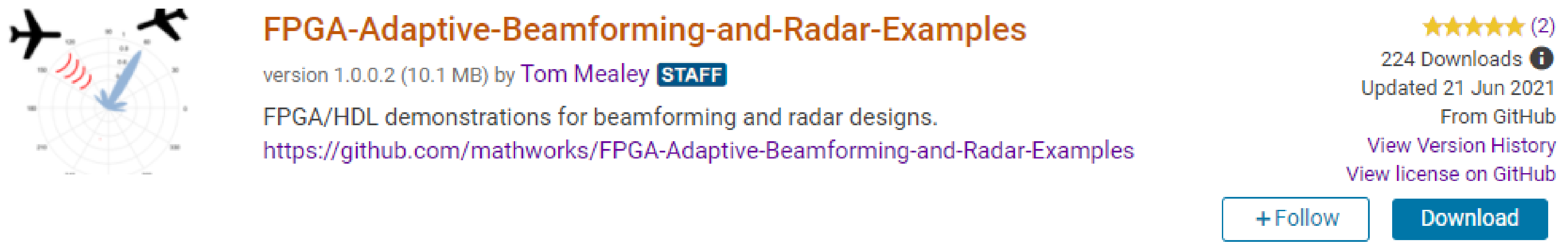


HDL Code Generation for Beamforming algorithms



Integrated Verification of HDL Code

Beamforming Demonstration



FPGA-Adaptive-Beamforming-and-Radar-Examples

version 1.0.0.2 (10.1 MB) by Tom Mealey STAFF

FPGA/HDL demonstrations for beamforming and radar designs.

https://github.com/mathworks/FPGA-Adaptive-Beamforming-and-Radar-Examples

★★★★★ (2)

224 Downloads

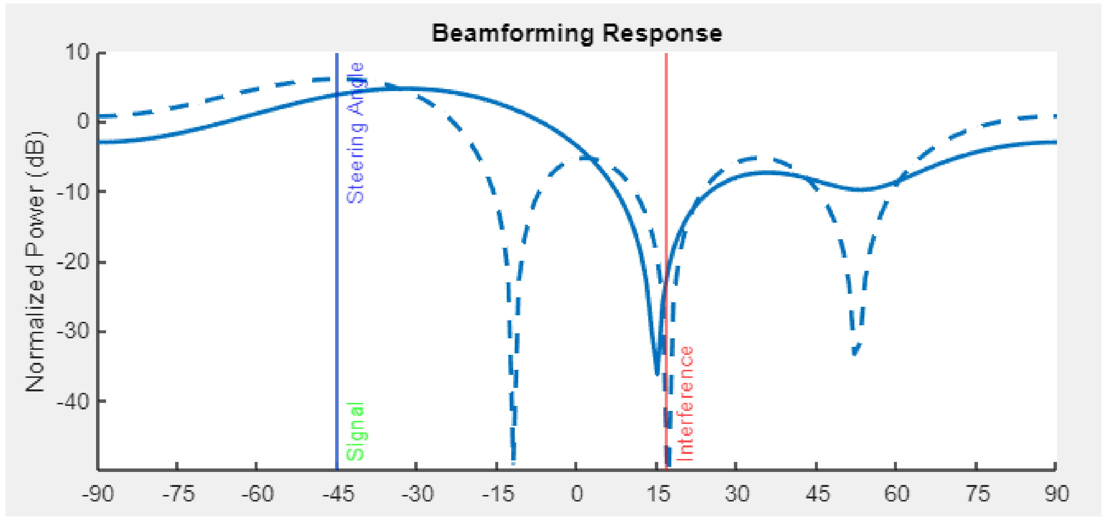
Updated 21 Jun 2021

From GitHub

View Version History

View license on GitHub

- ZCU111 RFSoC Adaptive Beamformer demo for 4x4 matrix solve for 4 channel ADC/DAC
- Places nulls in interference locations and maximizes beam pattern for steering direction
- Interactively steer angles for interference and beam pattern at run-time

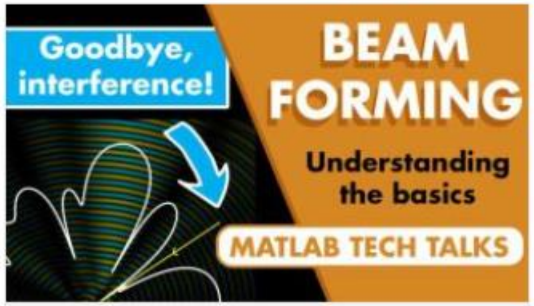


Download from File Exchange

Learn more about Phased Arrays and Beamforming basics

What Are Phased Arrays? (17:35)

Phased arrays are multiple sensors that act together to produce a desired sensor pattern and can be steered electronically simply by adjusting the phase of the signals to each individual element.



An introduction to Beamforming (13:57)

This video shows how adjusting the gain and phase unevenly to each element in an array provides a lot more flexibility in shaping what that beam looks like and opens up the possibility of adaptive beamforming.

Why multichannel beamforming is useful for wireless communication (13:14)

This video covers some of the reasons why multichannel beamforming is required to overcome the problems that we face with modern communication systems like 5G and WiFi.

Why Digital Beamforming Is Useful for Radar (13:07)

Learn how you can use digital beamformers to improve the performance and functions of radar systems.

Understanding Phased Array Systems and Beamforming

Learn More

- Visit MATLAB for FPGA, ASIC, and SoC Development solution page

MATLAB for FPGA, ASIC, and SoC Development

Automate your workflow — from algorithm development to hardware design and verification

- Learn more about FPGA Design with MATLAB
  - HDL Coder Self-Guided Tutorial
  - HDL Coder Evaluation Reference Guide

MATLAB EXPO

Thank you

MathWorks®