ANALOG DEVICES
AHEAD OF WHAT'S POSSIBLE™

MathWorks®

Software-Defined Radio with RF Systems on Module Workshop

TRAVIS COLLINS, PHD

ROBIN GETZ

ANALOG DEVICES

NOAM LEVINE

MATHWORKS

Agenda

	Day 1 – Tools and Physical Layer Algorithms	Day 2 – Hardware Implementation and MAC Layer
AM	• Introduction and demo • Workflow overview • **LAB**: IIO-Scope • Simulation, hardware evaluation with algorithms • **LAB:** MATLAB Streaming • Modem overview • **LAB**: Modem test harness • Discussion	• Day 1 review • Preparing for HDL and codegen • Deployment limitations and debugging • **LAB: Observable deployed designs** • Traditional HDL debugging • Discussion

PM	• Introduction to modeling with Simulink
	  • Simulation and testing
	  • Fixed-Point conversion and HDL capability
	• **LAB:** Fixed-Point conversion example
	• Discussion
		• MODEM demo on RF SOM Box
		• HSP vs Custom Reference designs
		  • Linux/HDL/BSP
		• **LAB:** TUN/TAP with OSC and debug registers
		• Summary and discussion
		  • Next steps and advanced topics

ANALOG DEVICES
AHEAD OF WHAT'S POSSIBLE™

MathWorks®

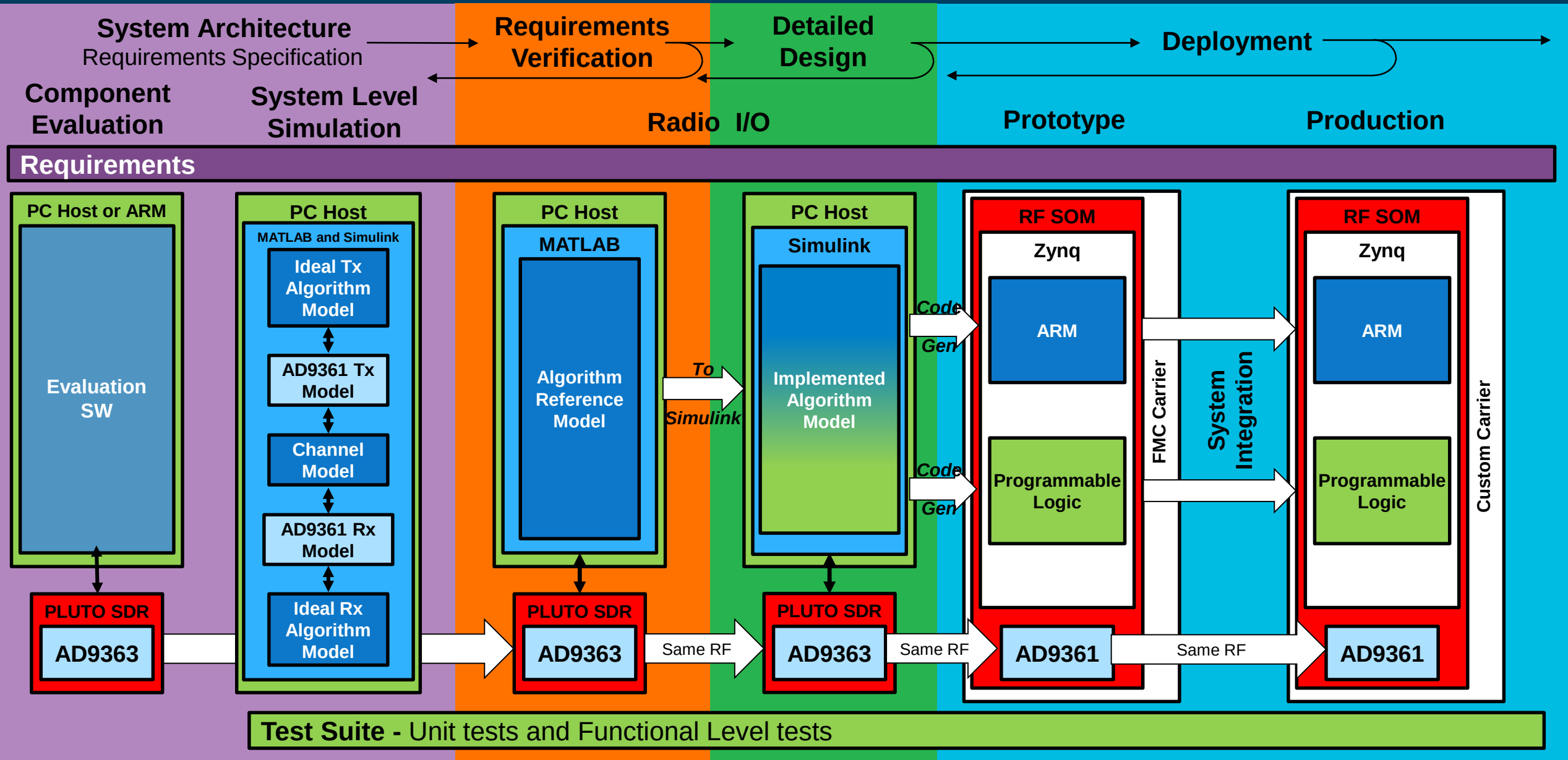
ANALOG DEVICES
AHEAD OF WHAT'S POSSIBLE™

MathWorks®

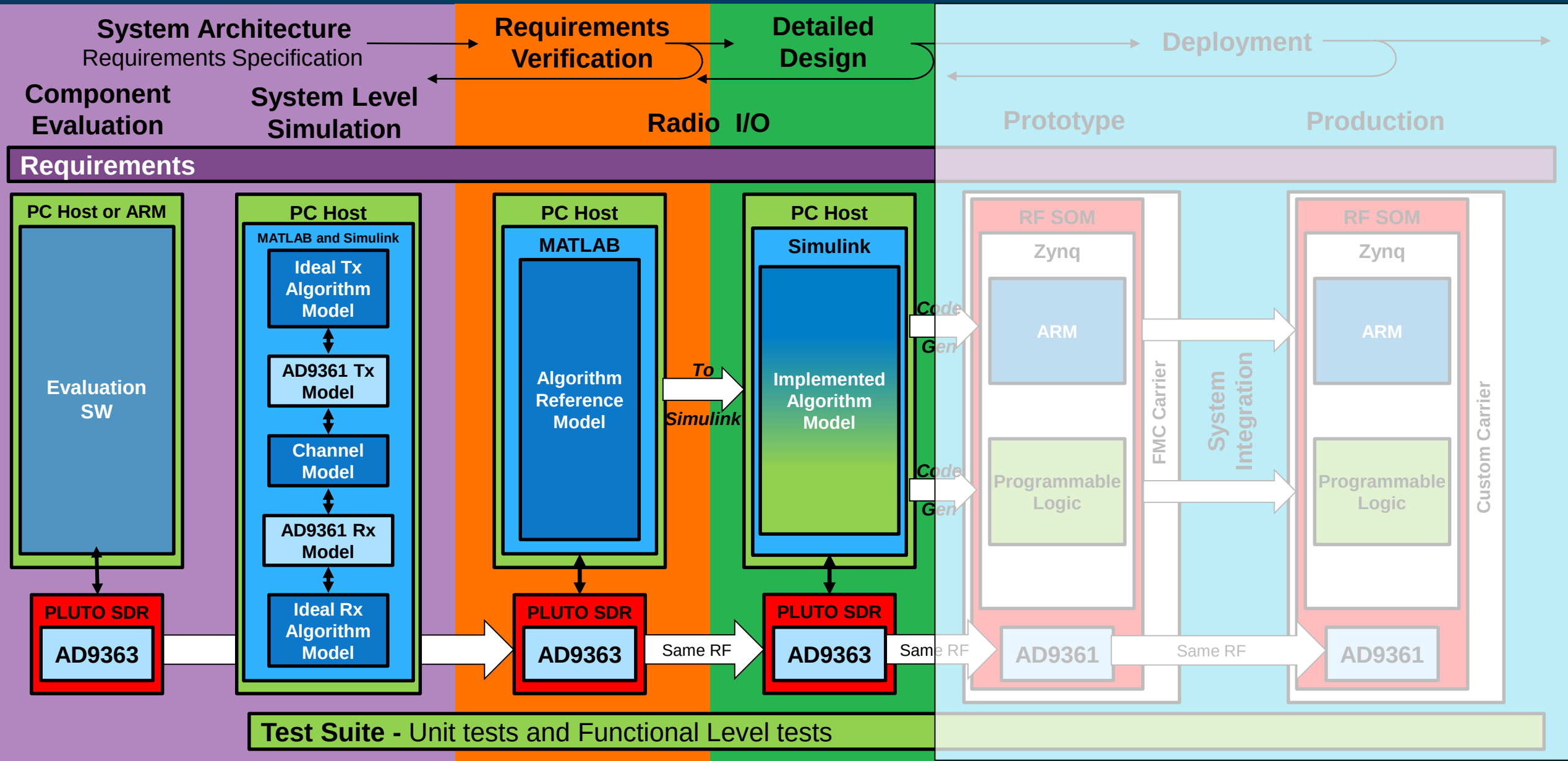
Day 2

NOAM LEVINE

Model-Based Design for SDR



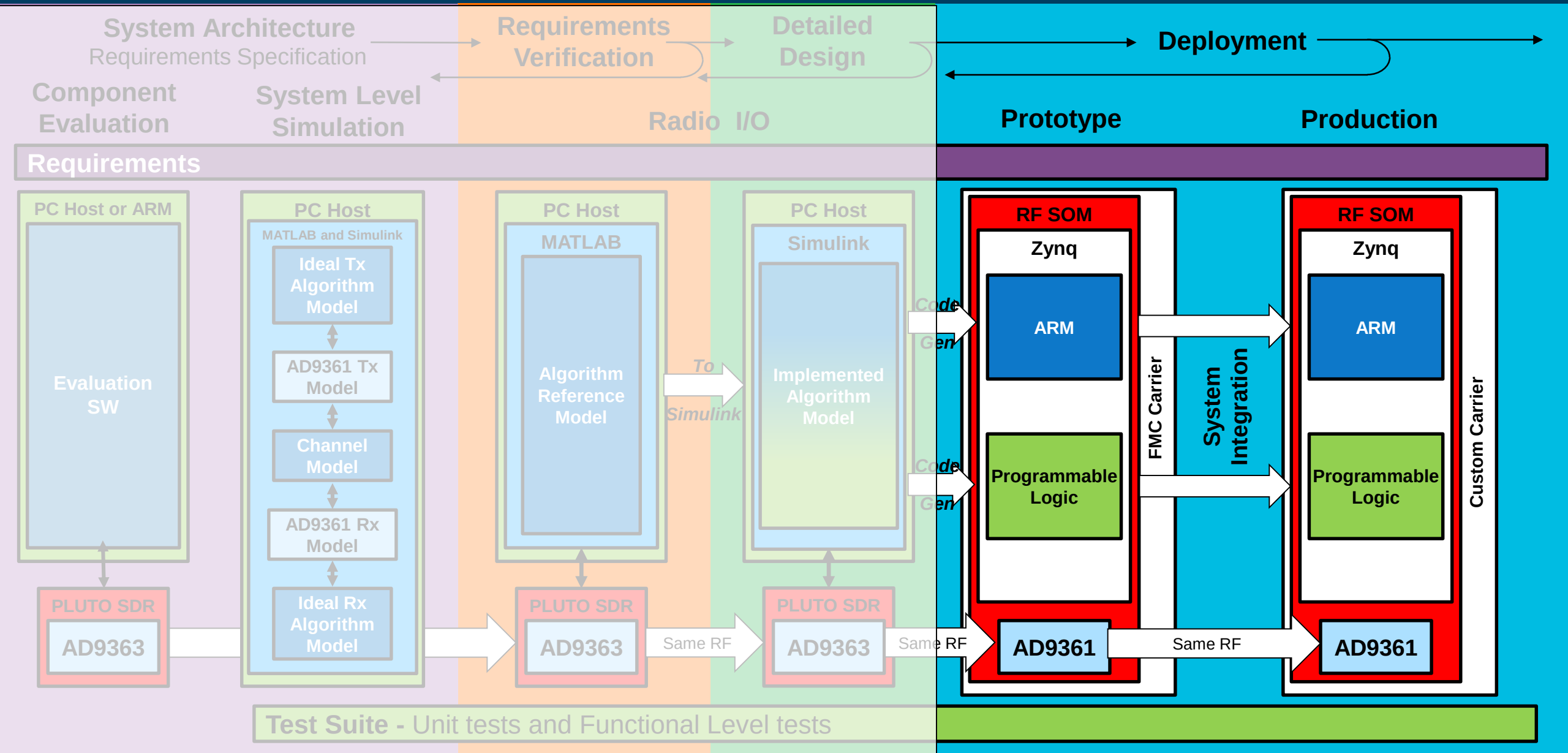
Model-Based Design for SDR



Day 1 Review

- Model-Based Design
- One Golden Reference, One Set of Requirements, One Test Infrastructure
- MATLAB
  - Algorithm development
  - Explore large data sets
  - Visualizations
- Simulink
  - Time-based simulation
  - Gateway to higher-level capabilities
    - Fixed-Point conversion
    - Code generation
- Example Modem Design:
  - MATLAB and Simulink focus on PHY+MAC layer development
  - Workflow process includes three designs:
    - MATLAB Floating-Point Reference Design
    - Simulink Floating-Point Scalar Model
    - Simulink Fixed-Point Scalar Model
  - Testing harness maintains requirement consistency across designs
- MATLAB and Simulink have many tools and features to help with fixed-point conversion process
- Testing early with real data saves time and money

Model-Based Design for SDR



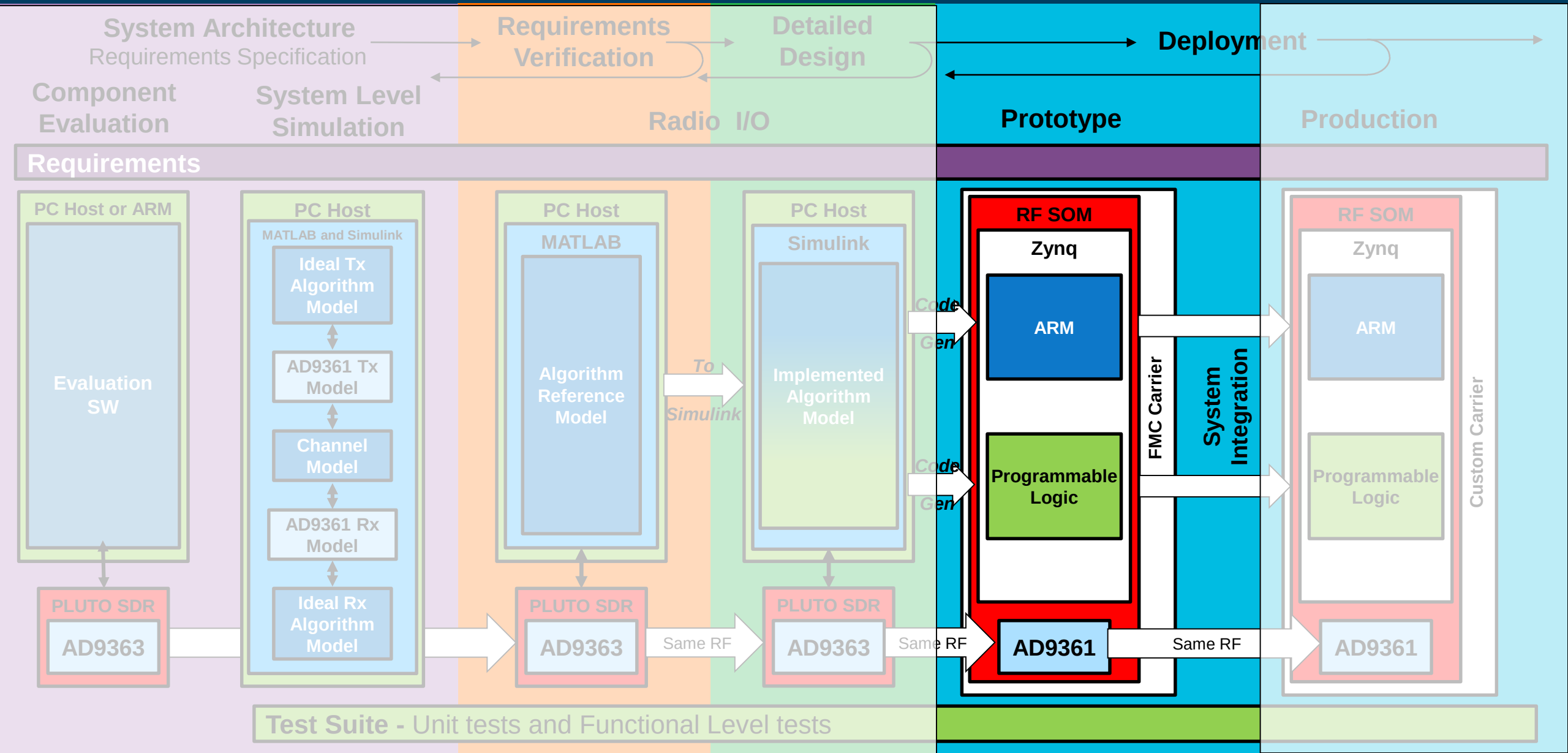
HDL Generation and Debugging

TRAVIS COLLINS, PHD

Lecture Outline

- Preparing models for HDL code generation and the process itself
  - Example: Changes made in Modem to meet requirements
- Integration of produced IP into RFSOM design
- Methods for debugging a deployed design

Model-Based Design for SDR



Model Progression

Software Components

MATLAB Floating-Point Model

Simulink Scalar Model

Simulink Fixed-Point Model

Production Interface

Software Flow

Design Tasks

Algorithm Performance Design

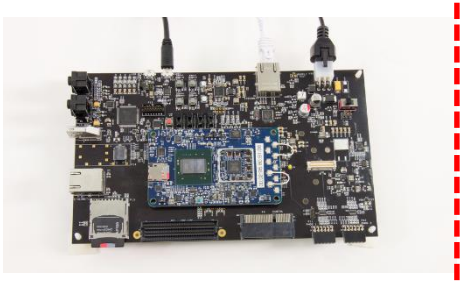
Algorithm System Level Design

HDL Deployment

Complete Stack Integration

Hardware Flow

Hardware Components



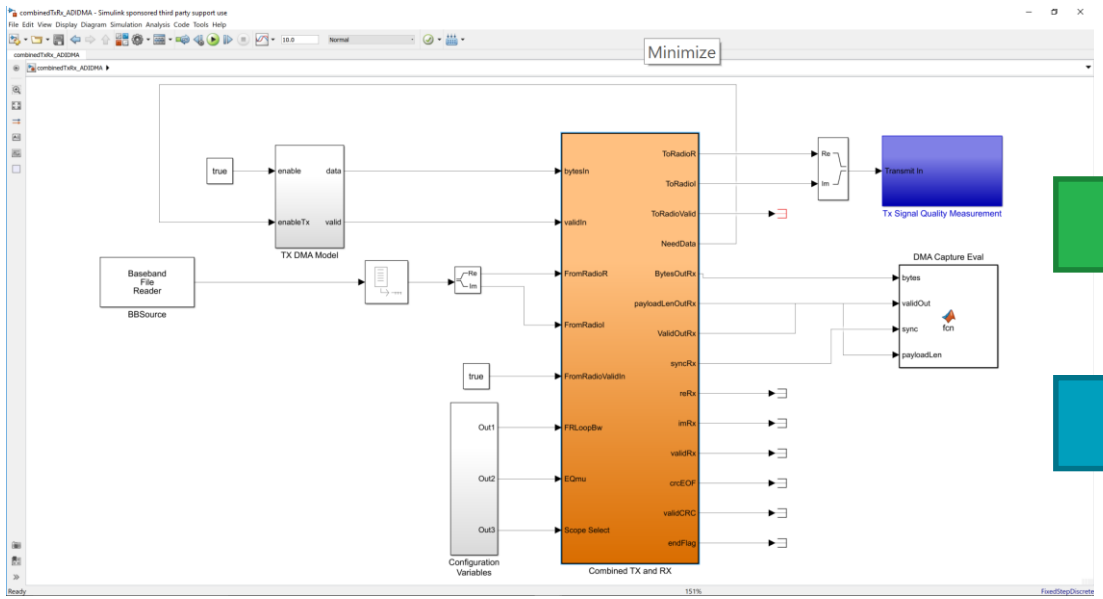
AHEAD OF WHAT'S POSSIBLE™

Works®

Instructor led Demo

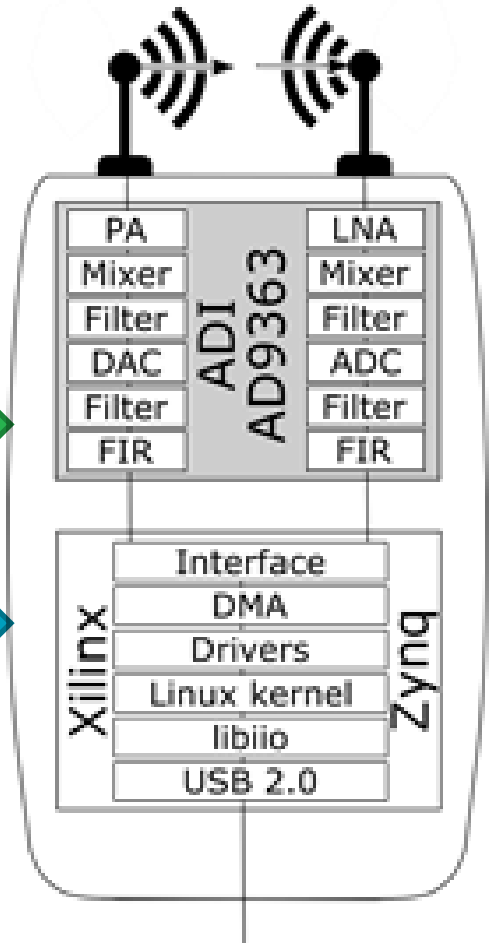
TETHERED / DEPLOYED MODEM

Getting My Model On Hardware



HDL-Coder

Embedded-Coder

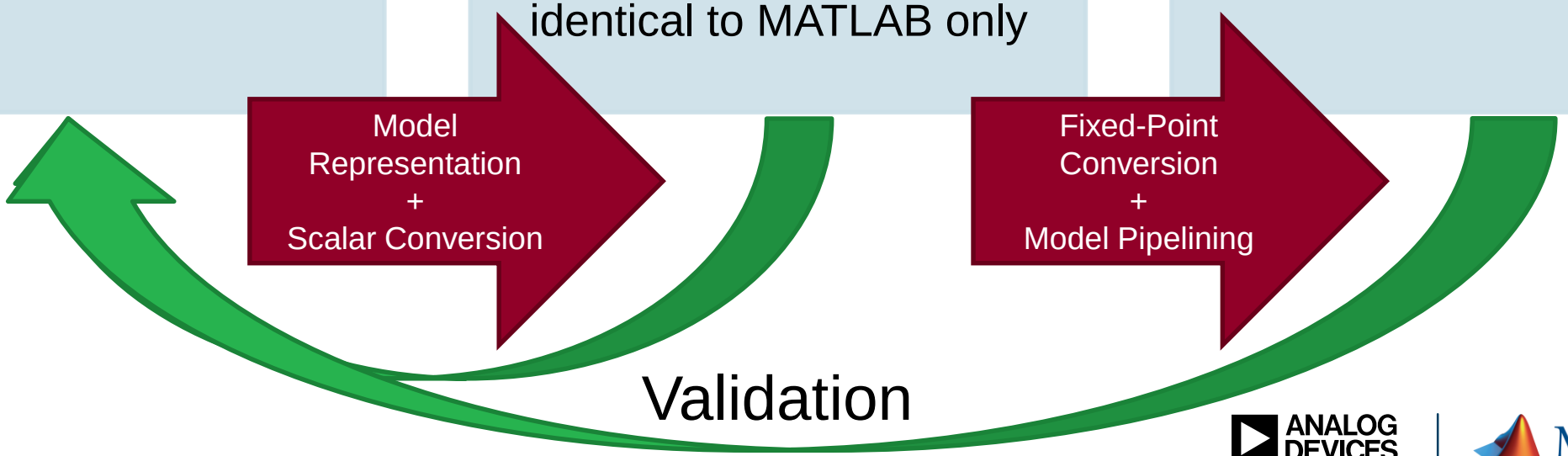


ANALOG DEVICES
AHEAD OF WHAT'S POSSIBLE™

MathWorks®

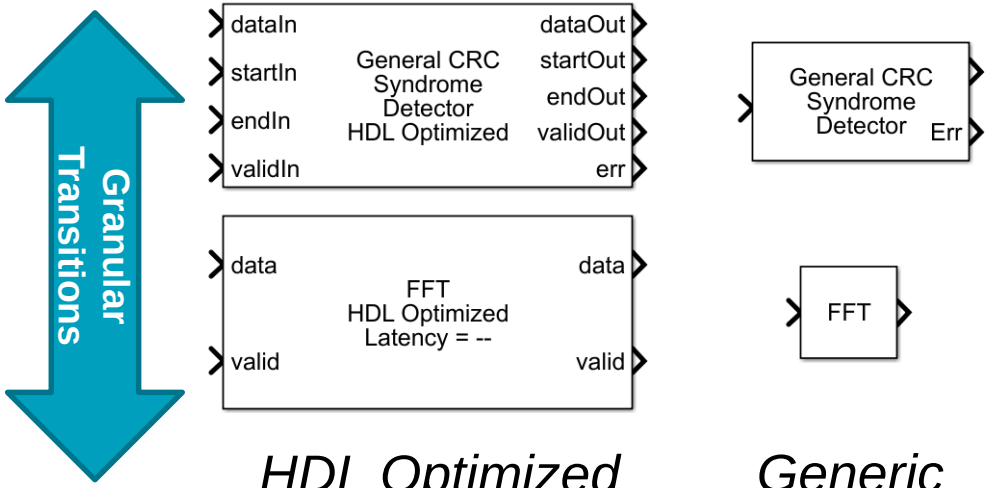
Design Process: Code (MATLAB) and Models (Simulink)

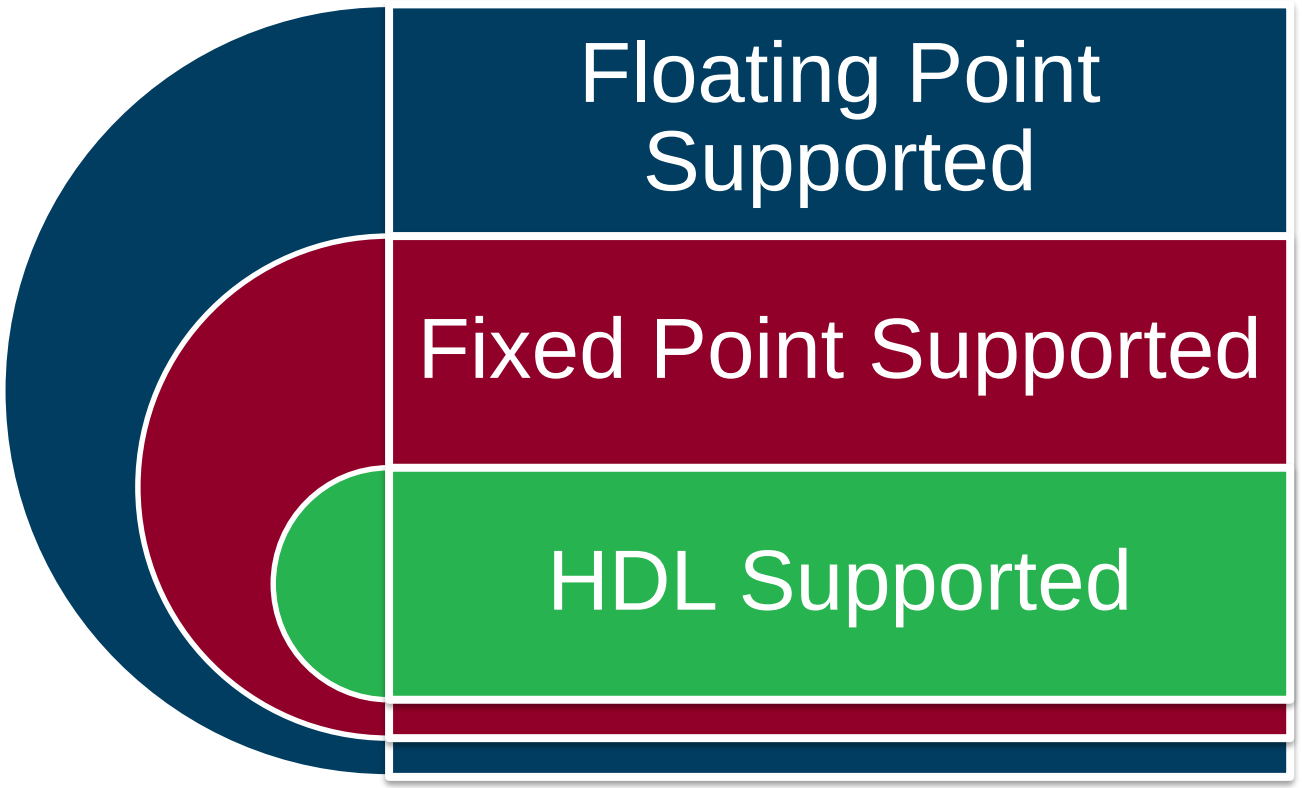
MATLAB Floating-Point Models	Simulink Scalar Model	Simulink Fixed-Point Model
• Fast running • Simple to debug • Best possible algorithmic performance	• *Scalar design* • Control signal propagation model • Algorithmic performance identical to MATLAB only	• *Fixed-Point HDL Capable* • Meets test points for performance validation



Tool Constraints and Design Evolution

- Implementation options
  - Many built in library blocks
  - Components can be built from atomic parts
  - *MATLAB Function Blocks can generate HDL*
- Read documentation on built-in blocks
  - Understand differences between:
    - HDL Optimized IP and
    - Simulation IP

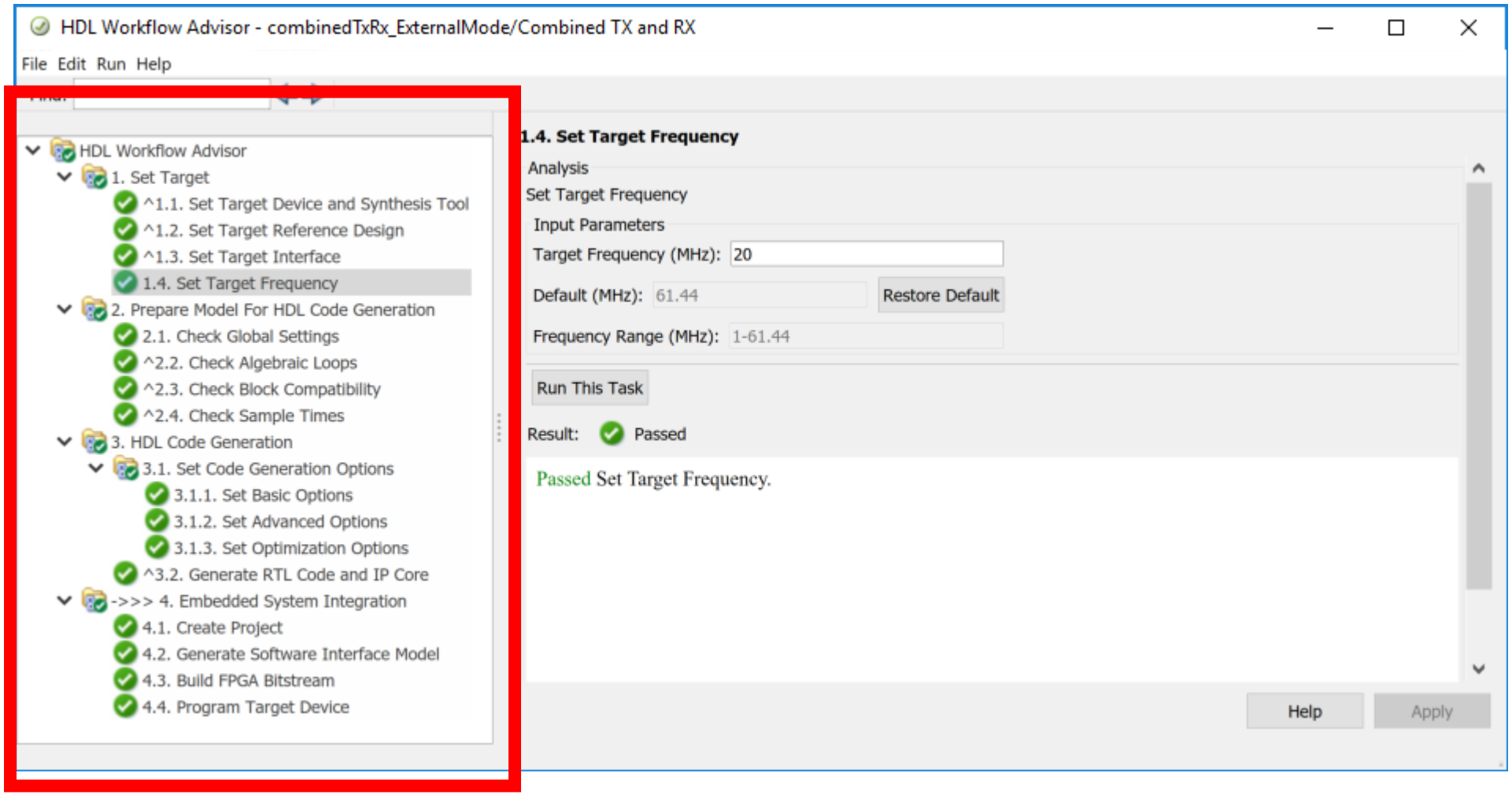




ANALOG DEVICES AHEAD OF WHAT'S POSSIBLE™ | MathWorks®

Where are we in the design process?

- Design is converted to fixed point
- Design is passing tests based on system requirements
- Now what?
  - A few more checks



ANALOG DEVICES
AHEAD OF WHAT'S POSSIBLE™

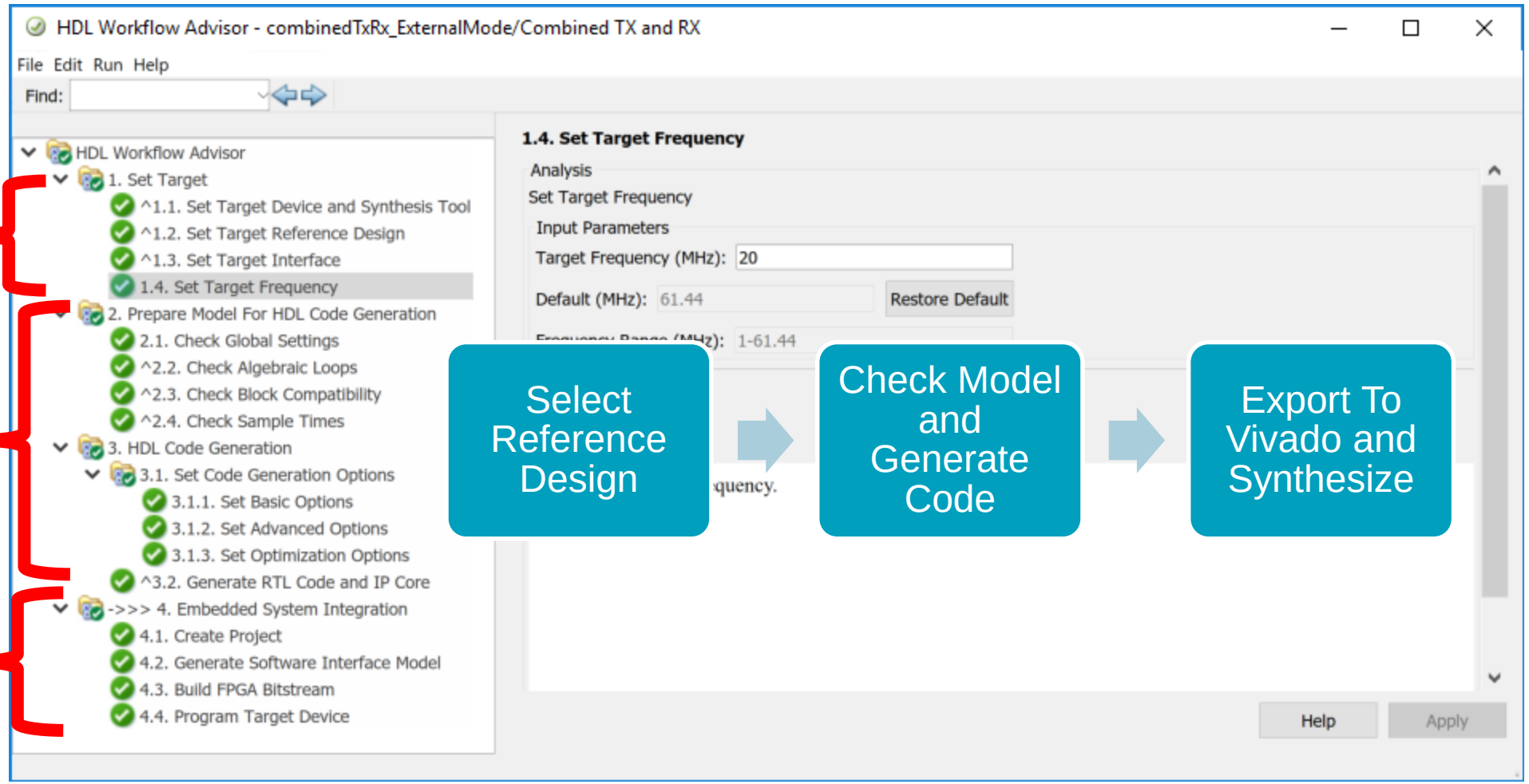
MathWorks®

Workflow Advisor Outline

Design Constrains

Code generation checks and optimizations

Synthesis and Deployment



ANALOG DEVICES
AHEAD OF WHAT'S POSSIBLE™

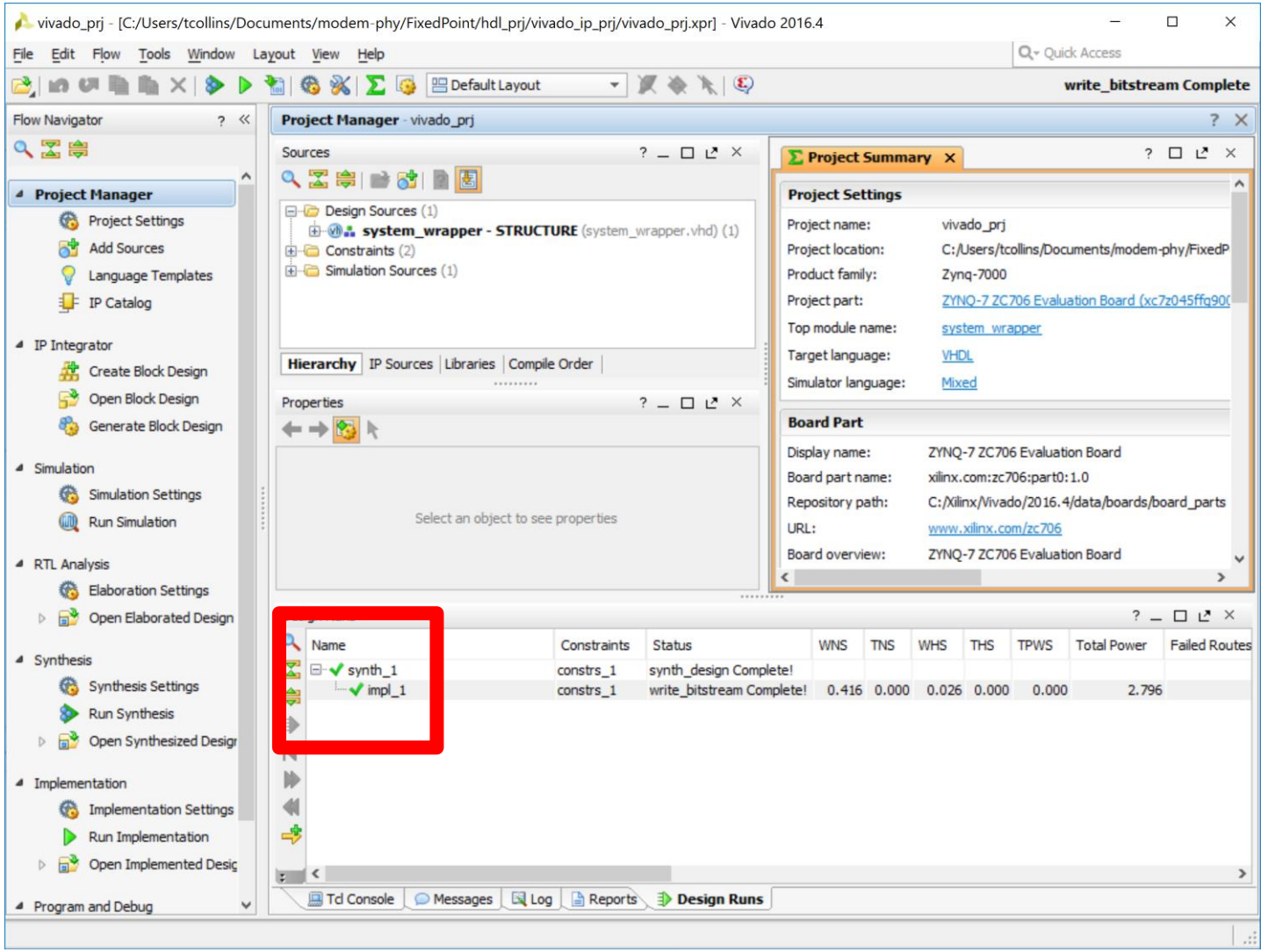
MathWorks®

ANALOG DEVICES
AHEAD OF WHAT'S POSSIBLE™

HDL Workflow Advisor Demo

Generated Code Expectations

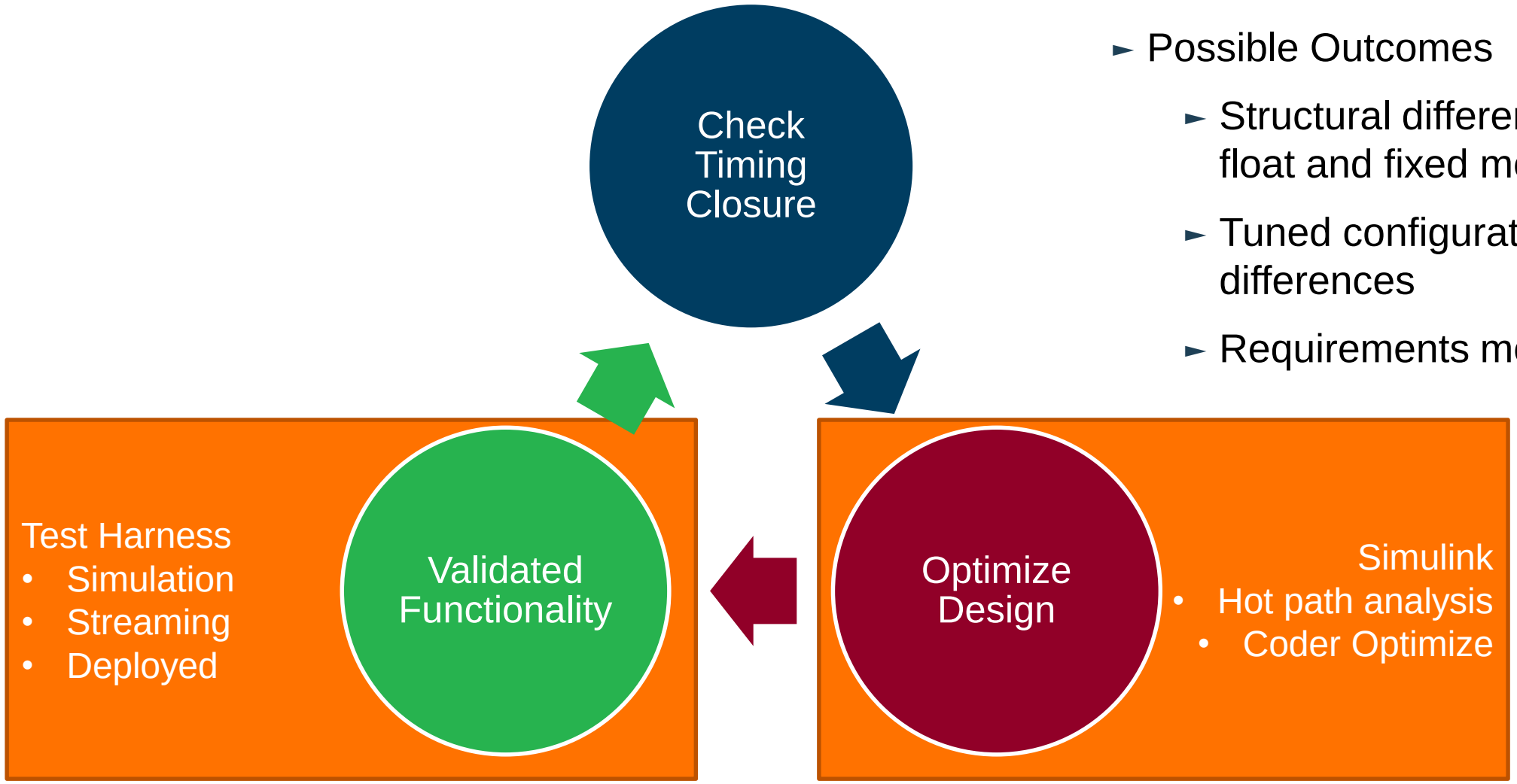
- HDL Code
  - Rate requirements for generated IP
  - Code is not guaranteed to meet timing closure
  - Code is readable and editable
  - Code is back annotated to model
- C/C++ Generated Code
  - Not for heavy DSP



ANALOG DEVICES
AHEAD OF WHAT'S POSSIBLE™

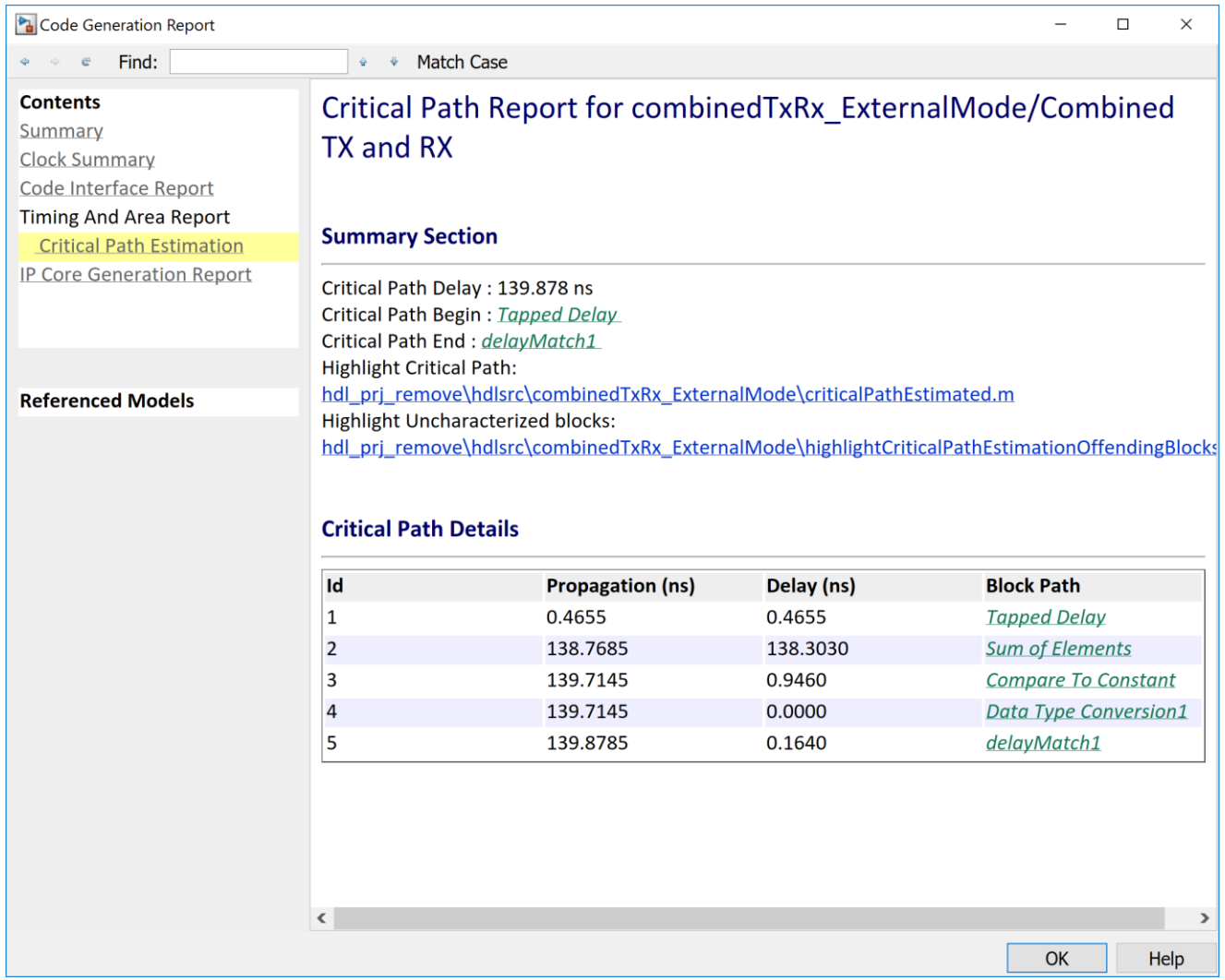
MathWorks®

Design Flow: Prototyping Stage



- Possible Outcomes
  - Structural differences between float and fixed models
  - Tuned configuration differences
  - Requirements modifications

Timing Reports: MATLAB and Vivado

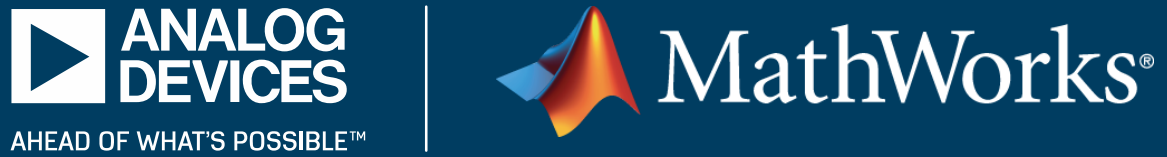


Id	Propagation (ns)	Delay (ns)	Block Path
1	0.4655	0.4655	Tapped Delay
2	138.7685	138.3030	Sum of Elements
3	139.7145	0.9460	Compare To Constant
4	139.7145	0.0000	Data Type Conversion1
5	139.8785	0.1640	delayMatch1

```
Max Delay Paths
--------------------------------------------------------------------------------
Slack (MET) :             12.297ns  (required time - arrival time)
  Source:                 system_i/CombinedT_ip_0/U0/u_CombinedT_ip_dut_ins
                            (rising edge-triggered cell DSP48E1 clocked by
  Destination:            system_i/CombinedT_ip_0/U0/u_CombinedT_ip_dut_ins
                            (rising edge-triggered cell FDCE clocked by dut
  Path Group:             dut_clk
  Path Type:              Setup (Max at Slow Process Corner)
  Requirement:            50.000ns  (dut_clk rise@50.000ns - dut_clk rise@0
  Data Path Delay:        37.519ns  (logic 20.735ns (55.266%)  route 16.784
  Logic Levels:           165  (CARRY4=137 DSP48E1=1 LUT2=2 LUT3=21 LUT4=4)
  Clock Path Skew:        -0.183ns (DCD - SCD + CPR)
    Destination Clock Delay (DCD):    1.233ns = ( 51.233 - 50.000 )
    Source Clock Delay      (SCD):    1.459ns
    Clock Pessimism Removal (CPR):    0.043ns
  Clock Uncertainty:      0.035ns  ((TSJ^2 + TIJ^2)^1/2 + DJ) / 2 + PE
    Total System Jitter     (TSJ):    0.071ns
    Total Input Jitter      (TIJ):    0.000ns
    Discrete Jitter          (DJ):    0.000ns
    Phase Error              (PE):    0.000ns

    Location             Delay type                Incr(ns)  Path(ns)    Ne
-------------------------------------------------------------------    --
                         (clock dut_clk rise edge)    0.000     0.000 r
    BUFGCTRL_X0Y0        BUFG                         0.000     0.000 r  sy
                         net (fo=27500, routed)       1.459     1.459    sy
    DSP48_X3Y94          DSP48E1                                     r  sy
-------------------------------------------------------------------    --
```


ANALOG DEVICES
AHEAD OF WHAT'S POSSIBLE™

MathWorks®



Coffee Break

ANALOG DEVICES
AHEAD OF WHAT'S POSSIBLE™

MathWorks®

Modem Design Decisions

MODEM DESIGN MODIFICATIONS FOR HDL GENERATION AND DEBUGGABILITY

AND

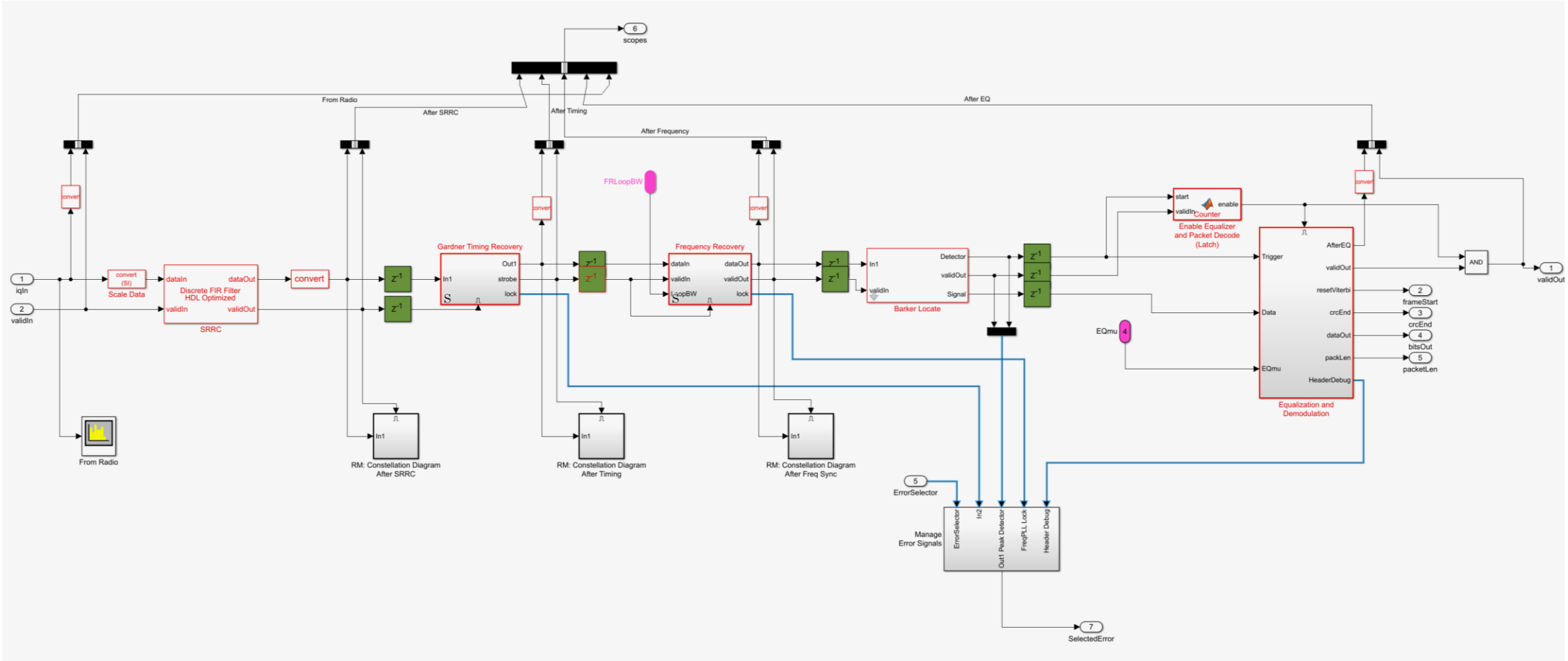
DESIGN INTEGRATION WITH REFERENCE DESIGN

TRAVIS COLLINS, PHD

Changes To Fixed-Point Model

- Why would we change our design?
  - HDL generation requirements not met
  - Not meeting timing closure
  - Design debuggability

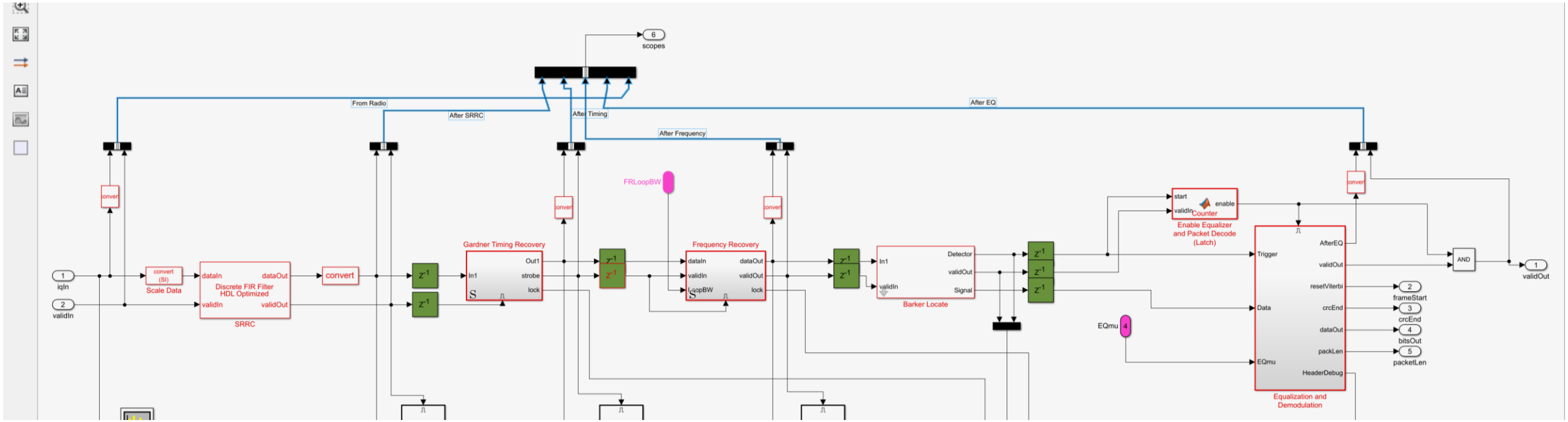
Debug Signals



ANALOG DEVICES
AHEAD OF WHAT'S POSSIBLE™

MathWorks®

Debug Scopes



ANALOG DEVICES
AHEAD OF WHAT'S POSSIBLE™

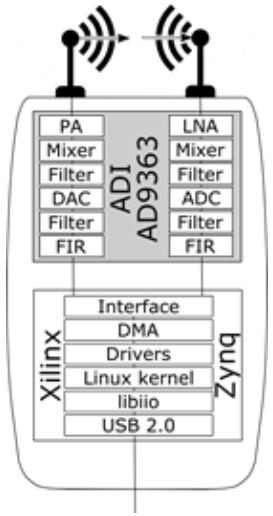
MathWorks®

Instructor lead Demo (Revisit)

TETHERED / DEPLOYED MODEM

Moving Toward FPGA Design and Design Methodoloy

- PLUTOSDR
  - 28k logic cells, 2.1 Mb Block RAM, 80 DSP Slices
  - USB ONLY



Larger FPGA Requirement

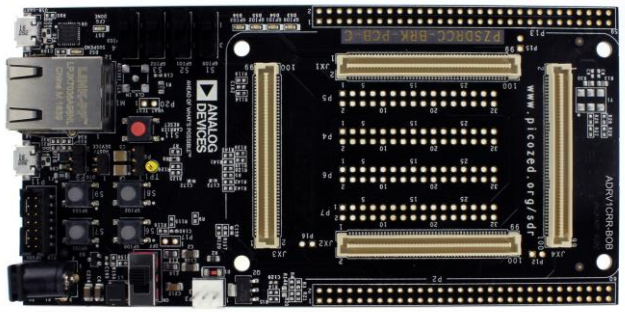
- RF-SOM
  - 85k logic cells, 4.9Mb Block RAM, 200 DSP Slices
  - 275k logic cells, 17.6Mb Block RAM, 900 DSP Slices

ANALOG DEVICES
AHEAD OF WHAT'S POSSIBLE™

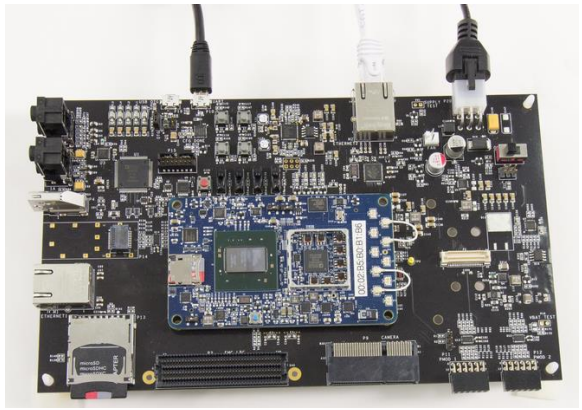
MathWorks®

RFSOM Hardware

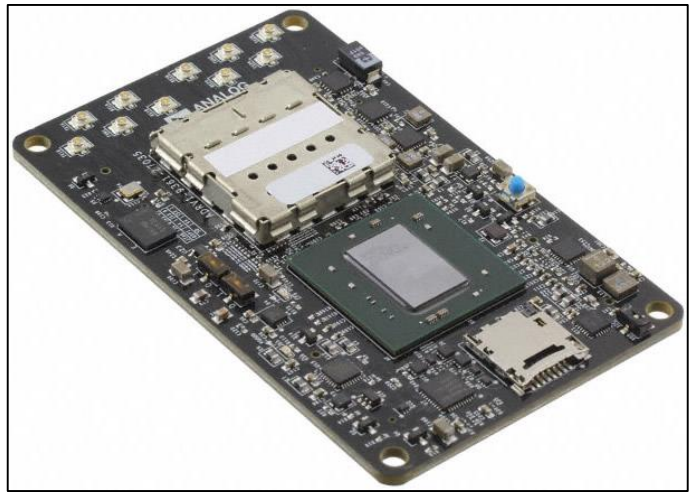
BOB



FMC



PCIe



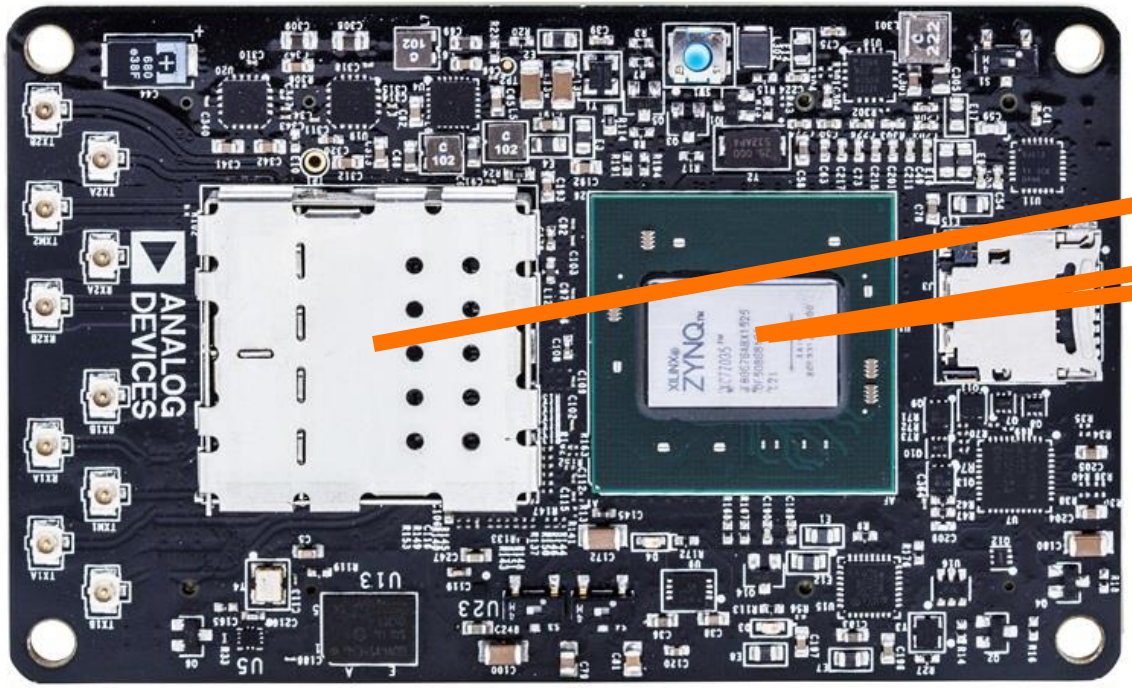
Same SOM

- RF-SOM Configurations
  - AD9361-Z7035
  - AD9364-Z7020
- Breakout boards
  - FMC
  - BOB

ANALOG DEVICES
AHEAD OF WHAT'S POSSIBLE™

MathWorks®

RF System-On-Module (SOM)



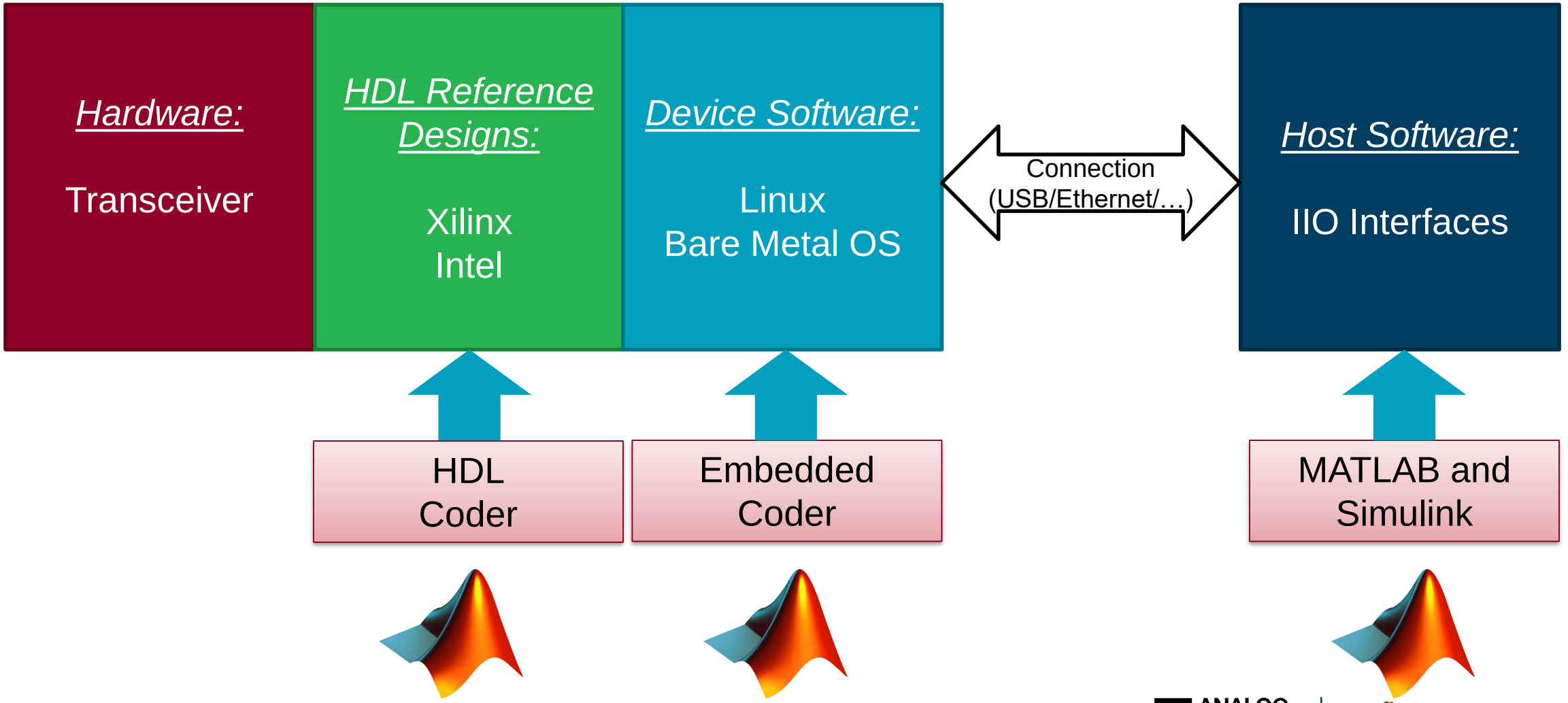
Hardware:	*HDL Reference Designs:*	*Device Software:*
Transceiver AD9361	Xilinx Intel	Linux Bare Metal OS

- ADI provides:
  - HDL reference designs
  - Linux and bare-metal drivers
  - Streaming API through IIO
  - All source at ***GitHub.com/AnalogDevicesInc***

ANALOG DEVICES
AHEAD OF WHAT'S POSSIBLE™

MathWorks®

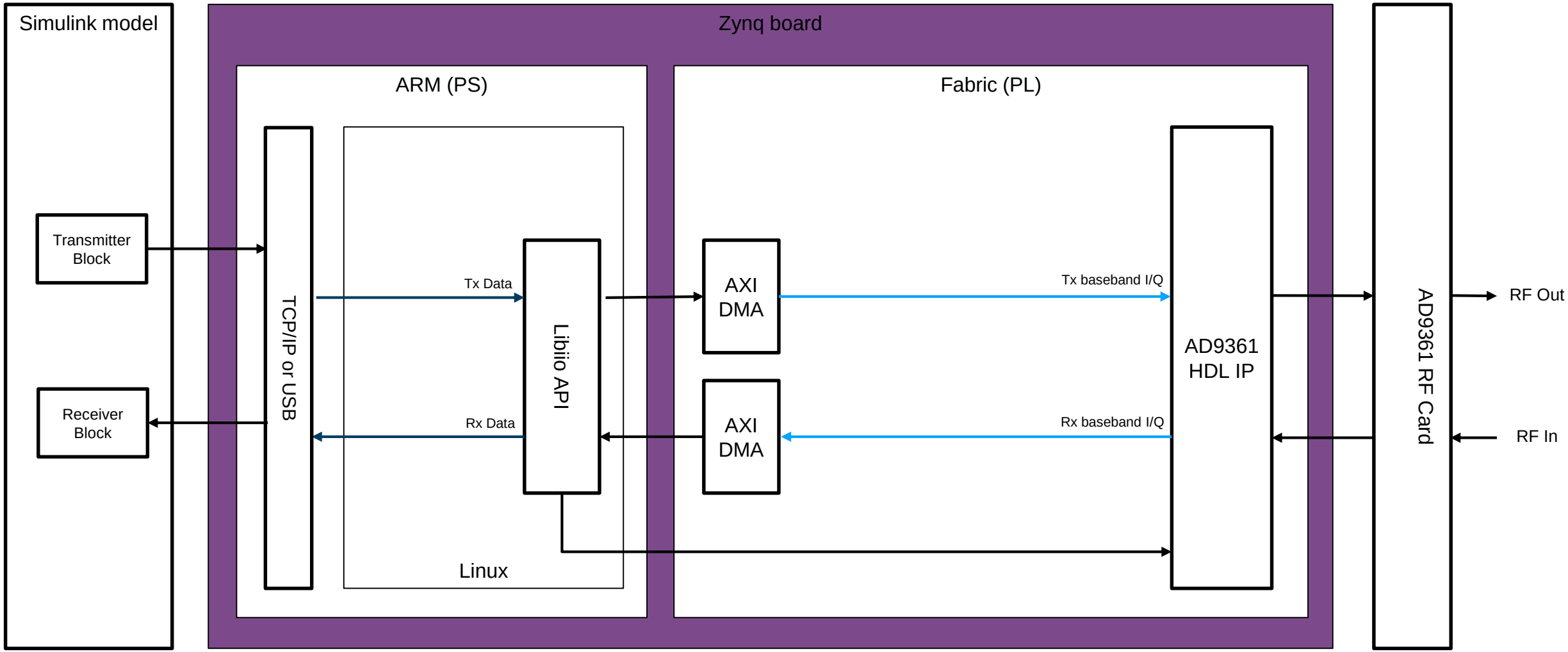
Hardware and Software Together: The Big Picture



ANALOG DEVICES
AHEAD OF WHAT'S POSSIBLE™

MathWorks®

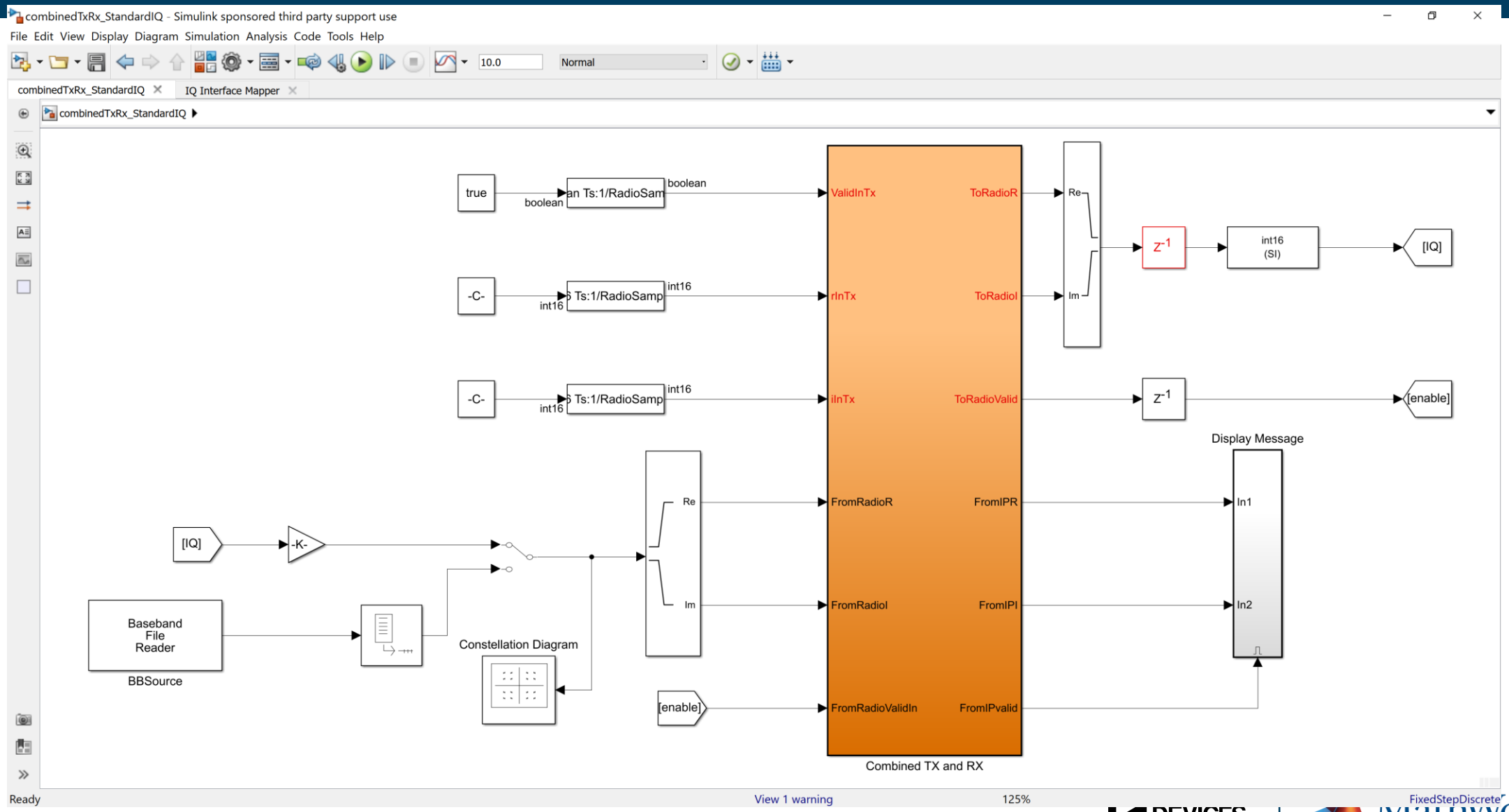
Detailed Data Flow: Reference Design



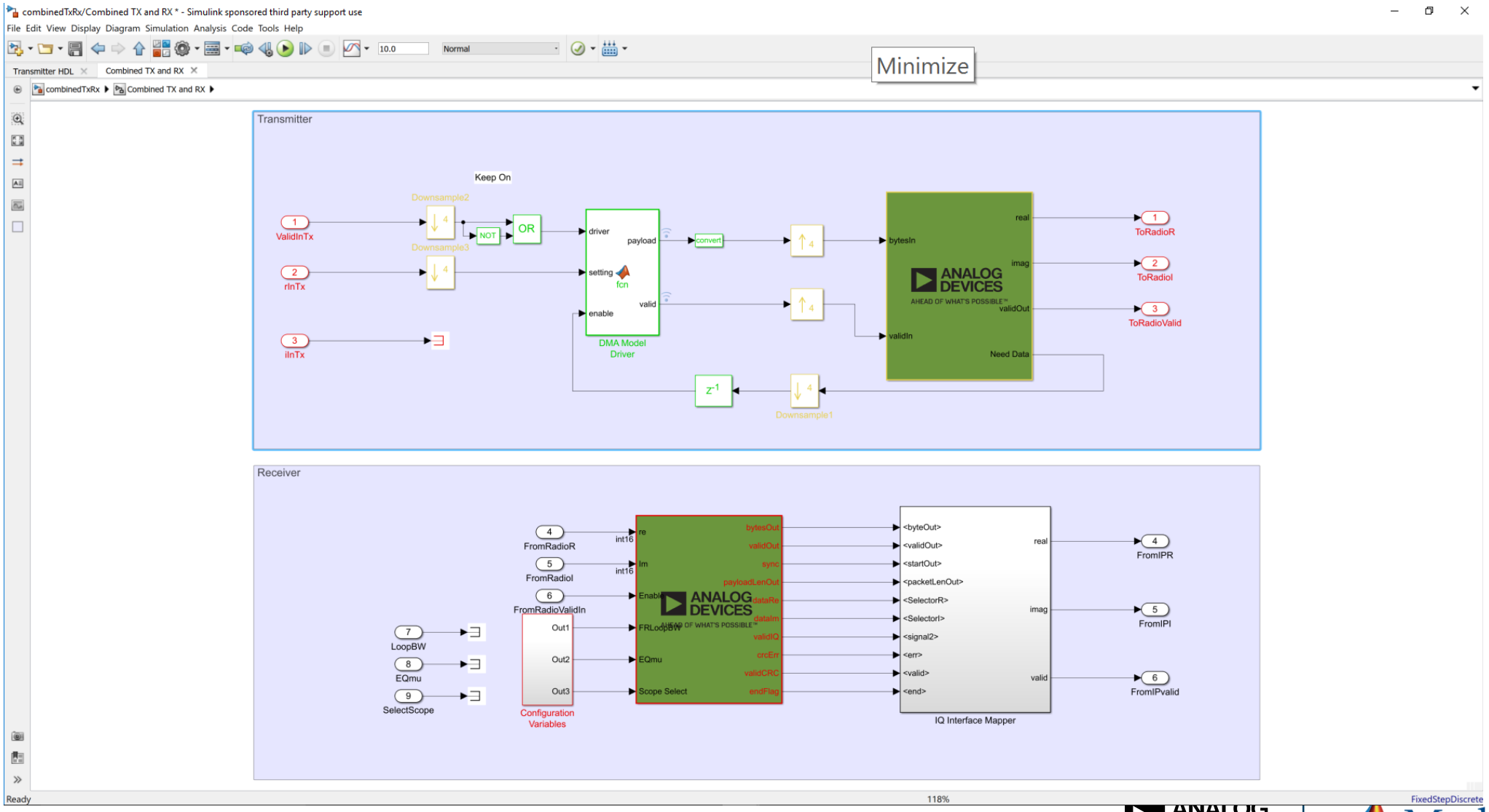
ANALOG DEVICES
AHEAD OF WHAT'S POSSIBLE™

MathWorks®

Main DSP Blocks of Modem Design



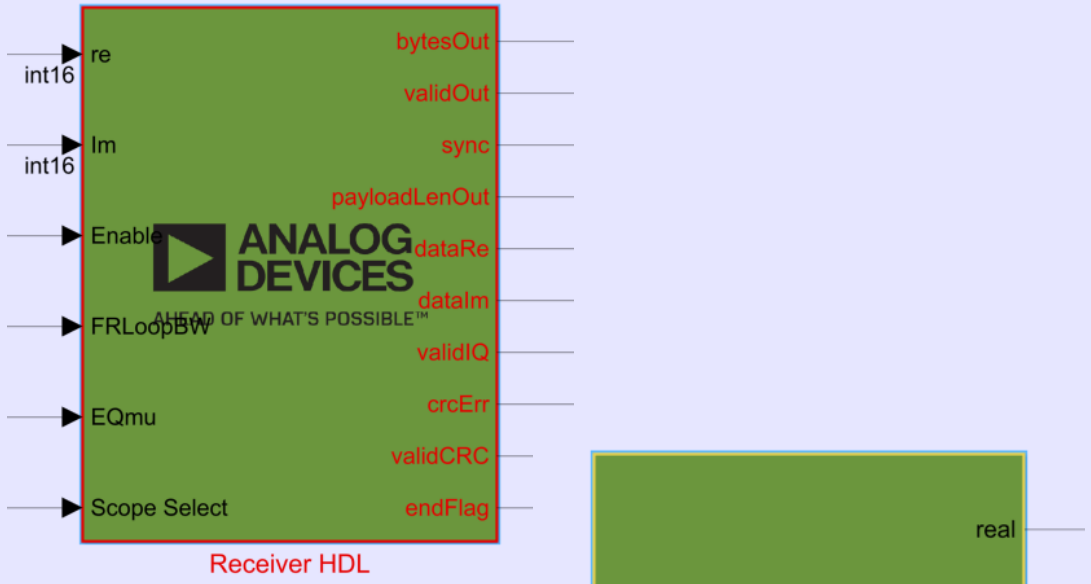
Example IQ Reuse



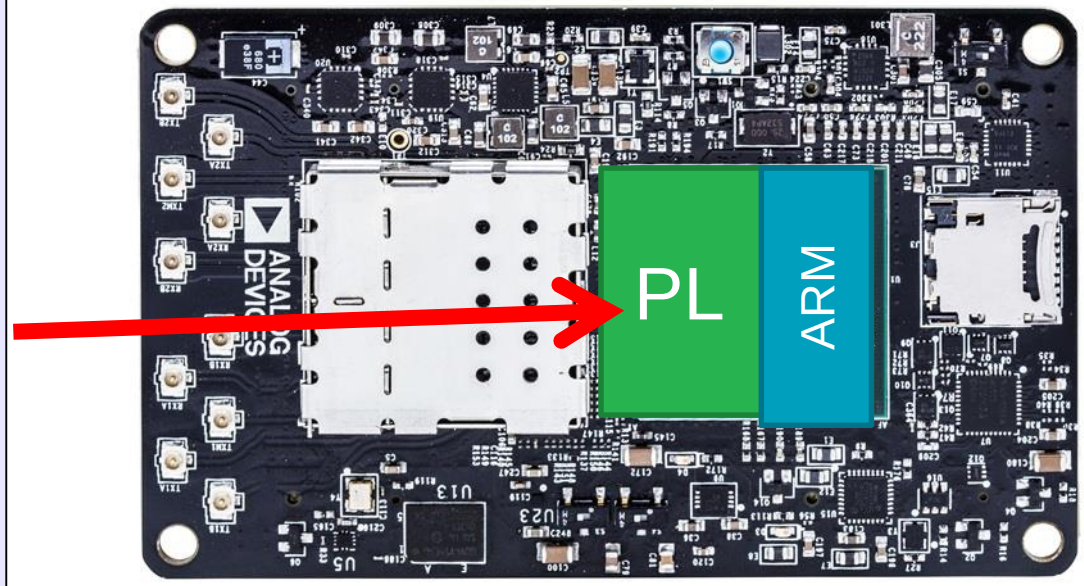
ANALOG DEVICES
AHEAD OF WHAT'S POSSIBLE™

MathWorks®

Mapping IP To Hardware



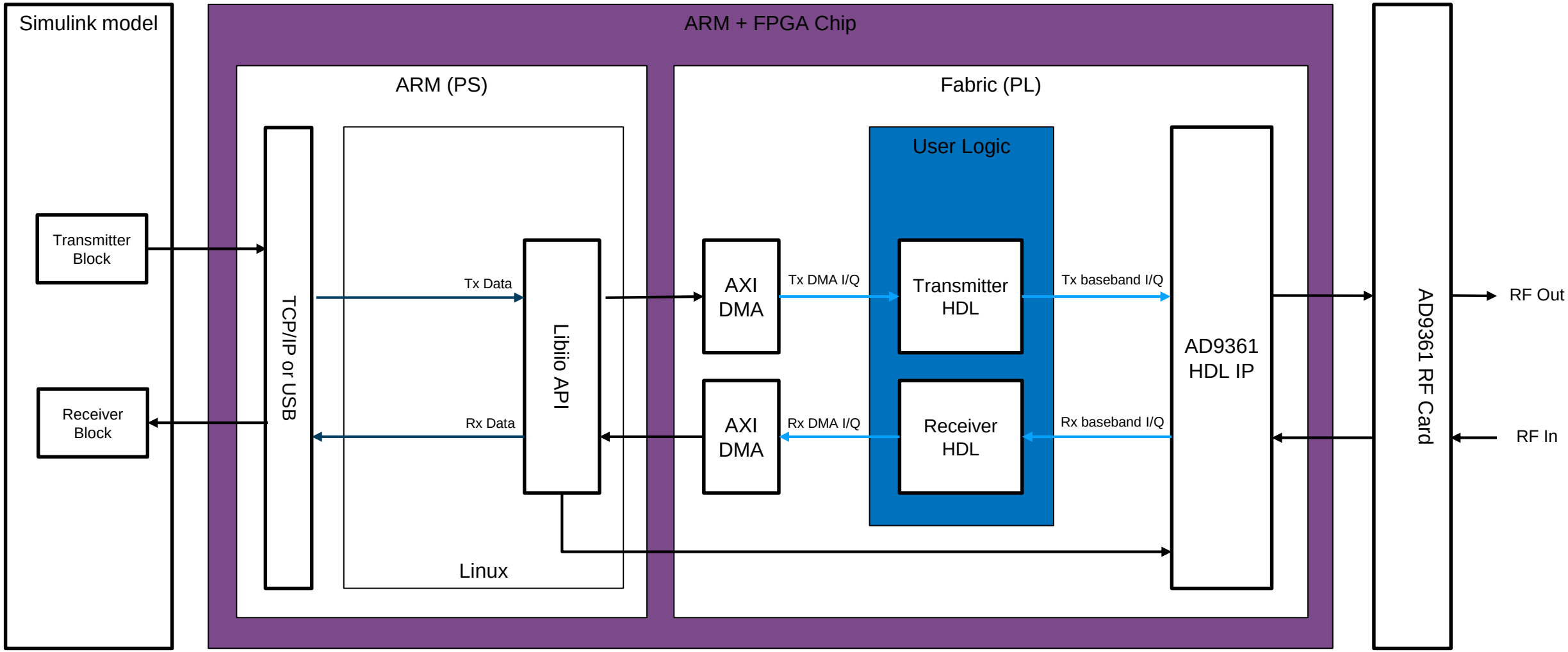
Larger Subsystem



Inserted into ADI HDL reference design

ANALOG DEVICES AHEAD OF WHAT'S POSSIBLE™ | MathWorks®

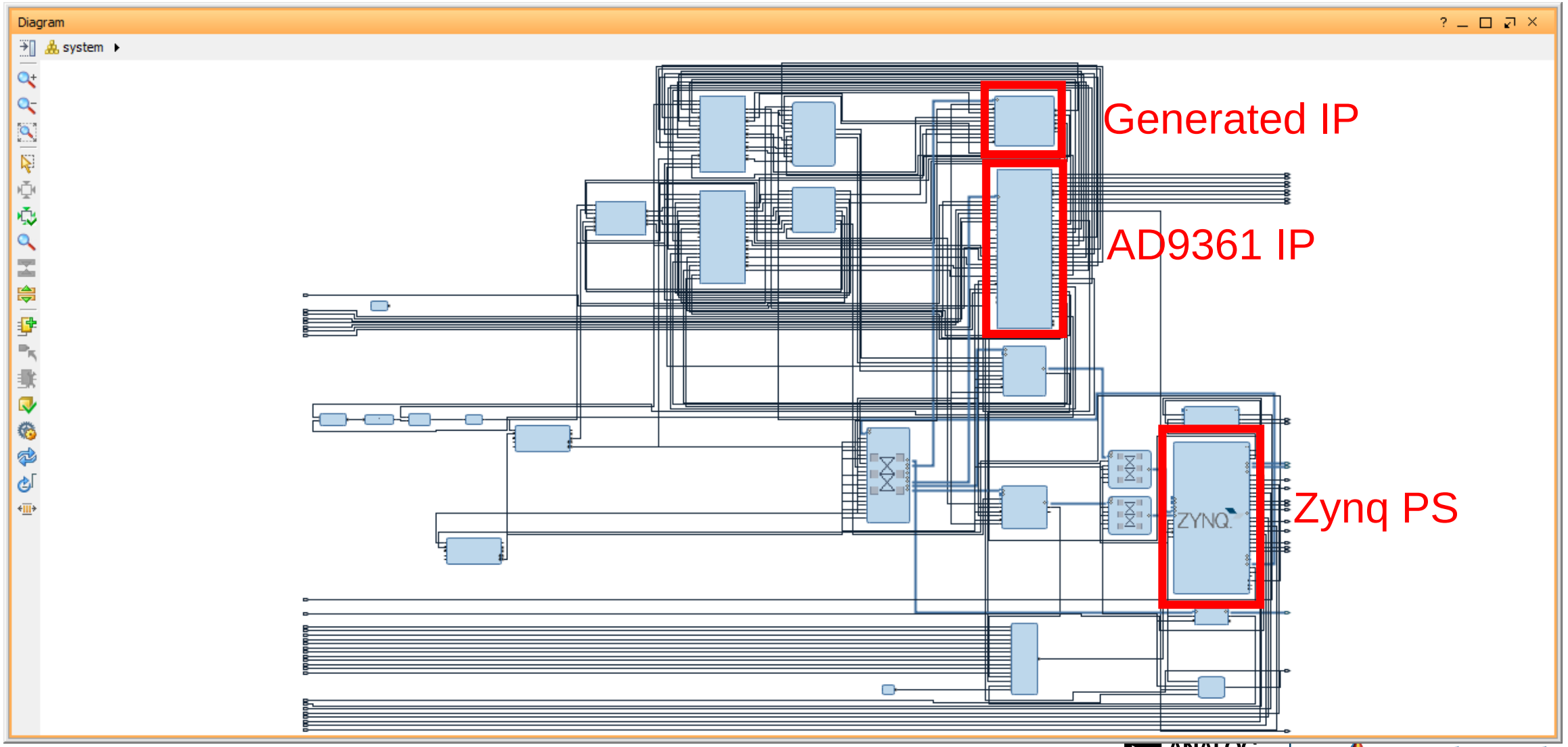
Detailed Data Flow: Adding IP To Model



ANALOG DEVICES
AHEAD OF WHAT'S POSSIBLE™

MathWorks®

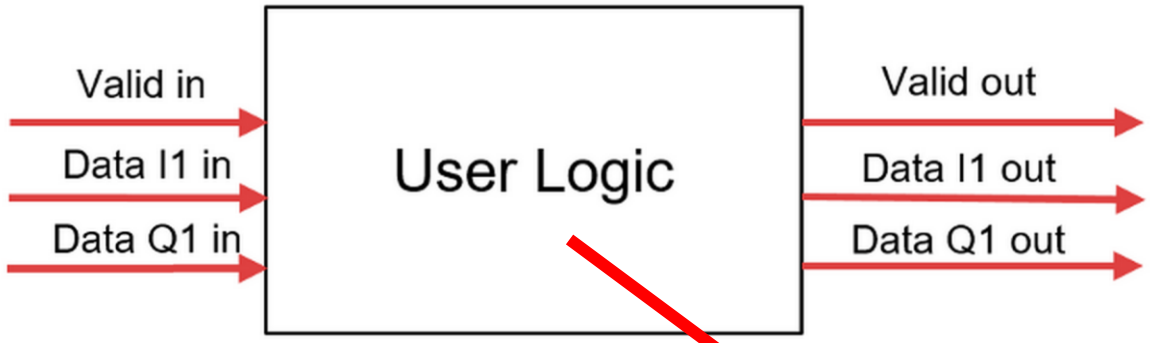
Example In Vivado Block Diagram



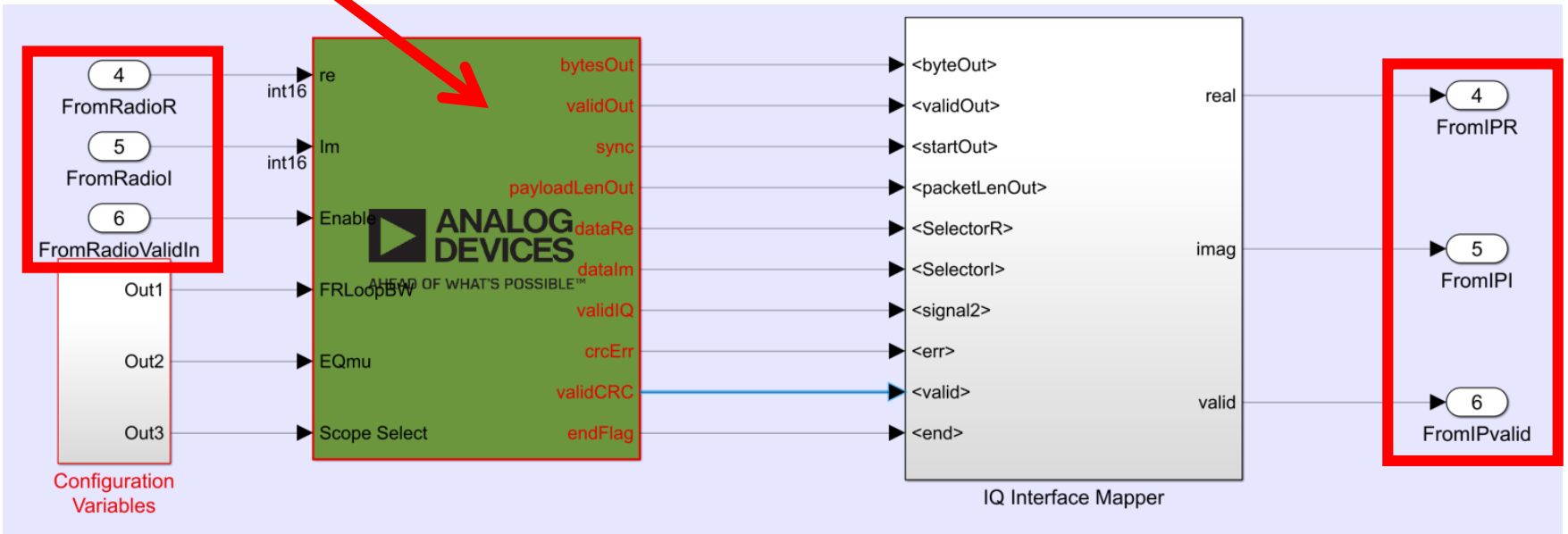
ANALOG DEVICES
AHEAD OF WHAT'S POSSIBLE™

MathWorks®

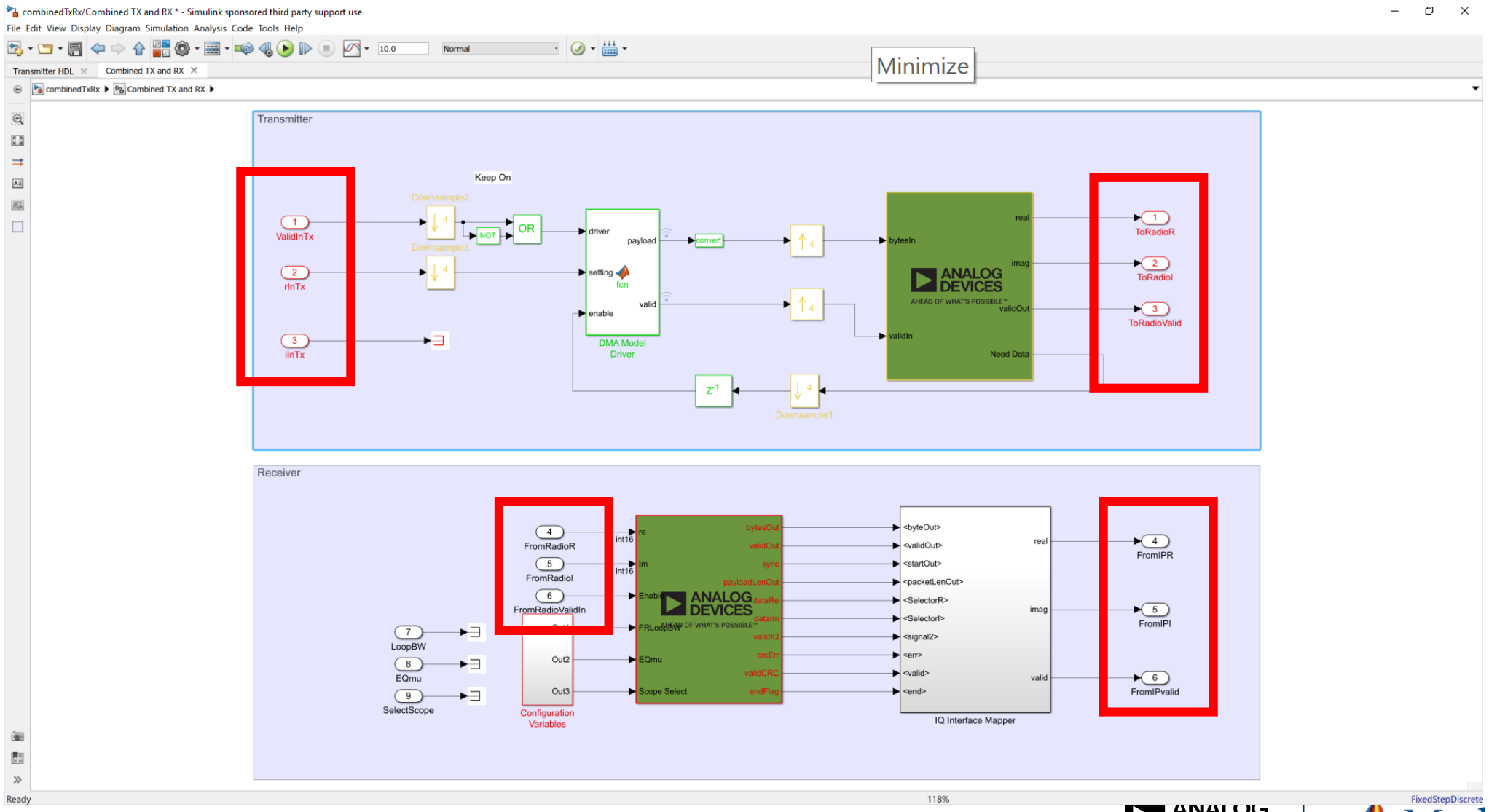
HSP: Default Reference Designs and Interfaces



- Default limitations
  - Reference designs interface rates must be at clock rate
  - By default I/Q lines are only input/output and must be populated



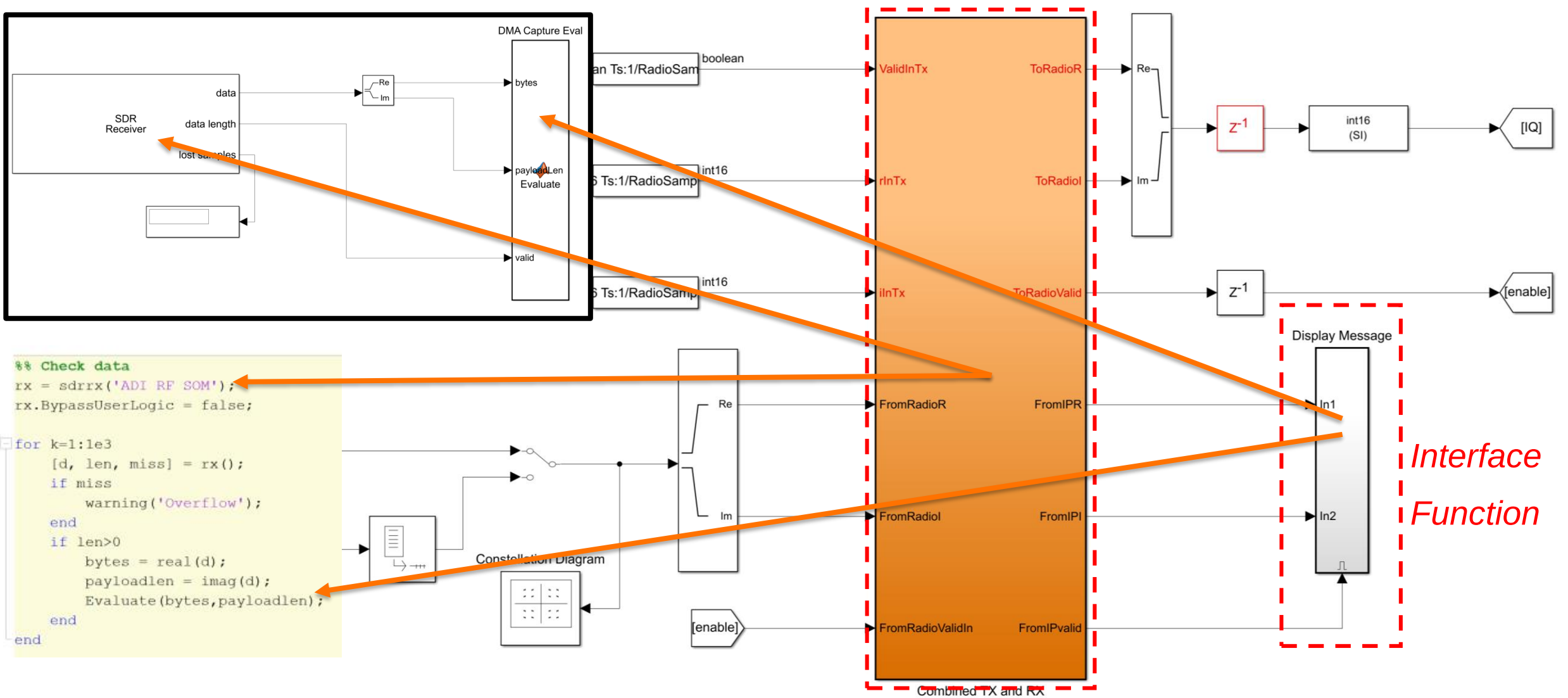
Example IQ Reuse



ANALOG DEVICES
AHEAD OF WHAT'S POSSIBLE™

MathWorks®

Back In Simulink Model

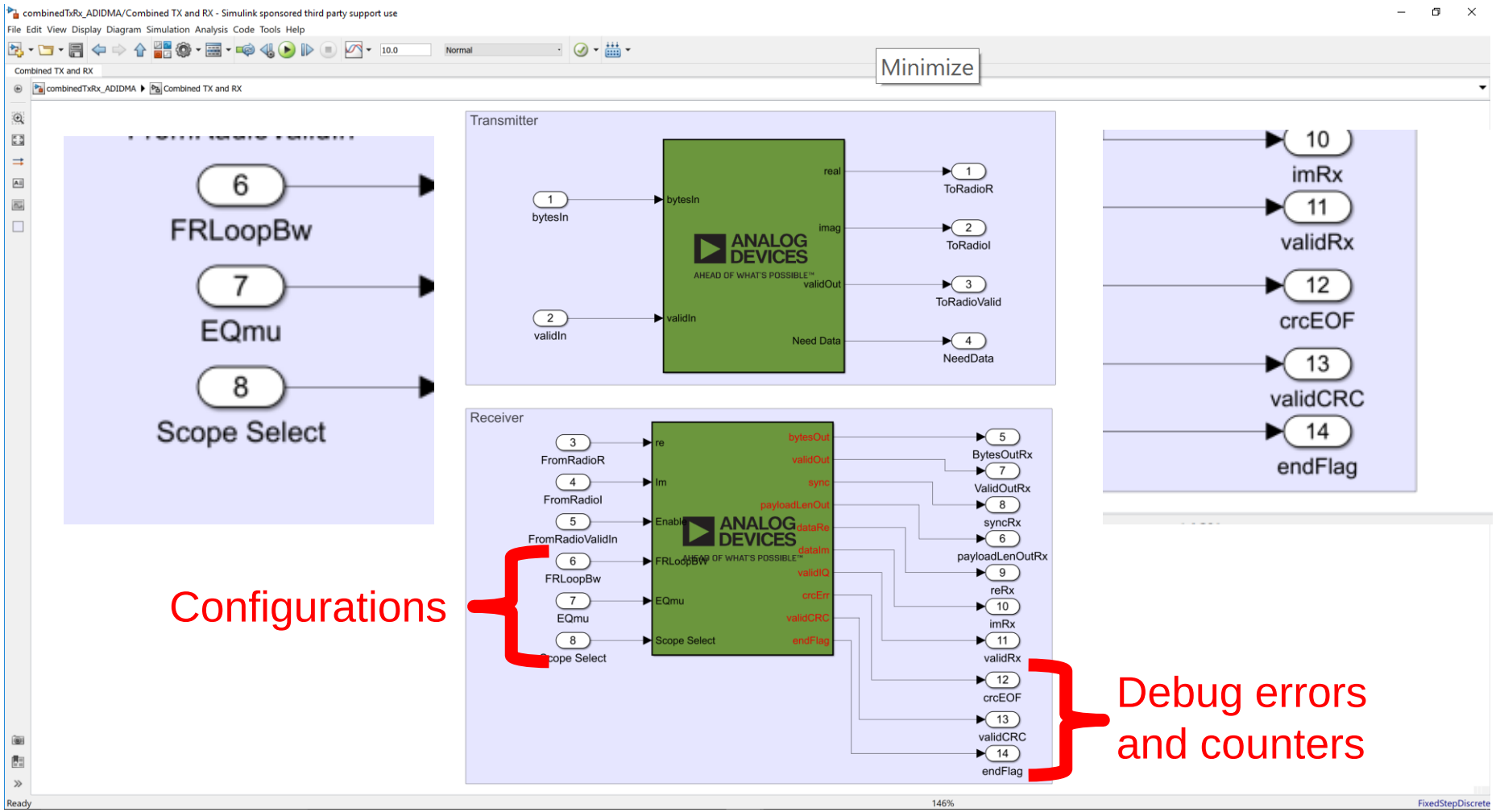


ANALOG DEVICES
AHEAD OF WHAT'S POSSIBLE™

MathWorks®

Lab 3 Part A: Standard IQ Reuse

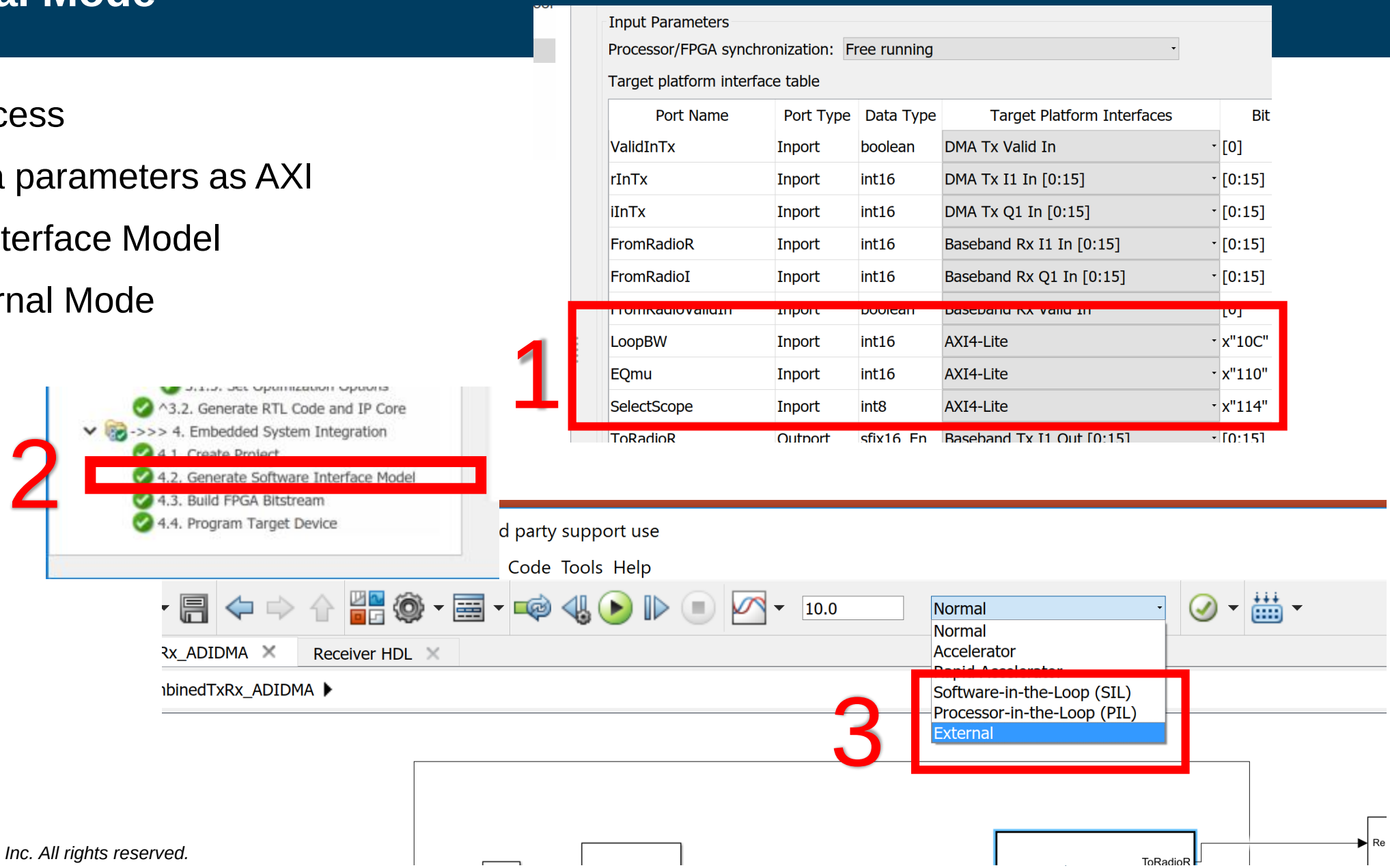
Main DSP Blocks of Modem Design



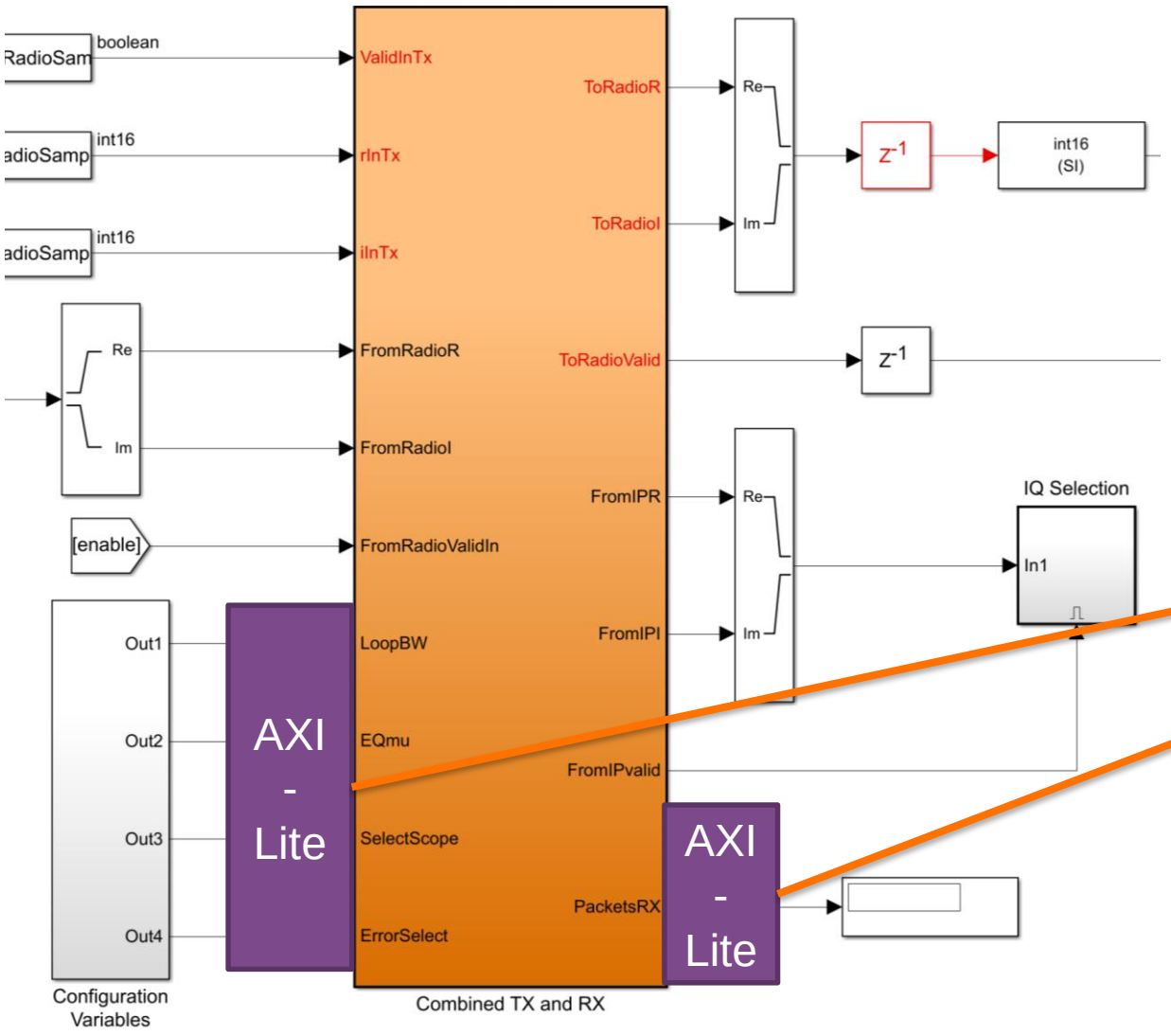
ANALOG DEVICES AHEAD OF WHAT'S POSSIBLE™ | MathWorks®

Use of External Mode

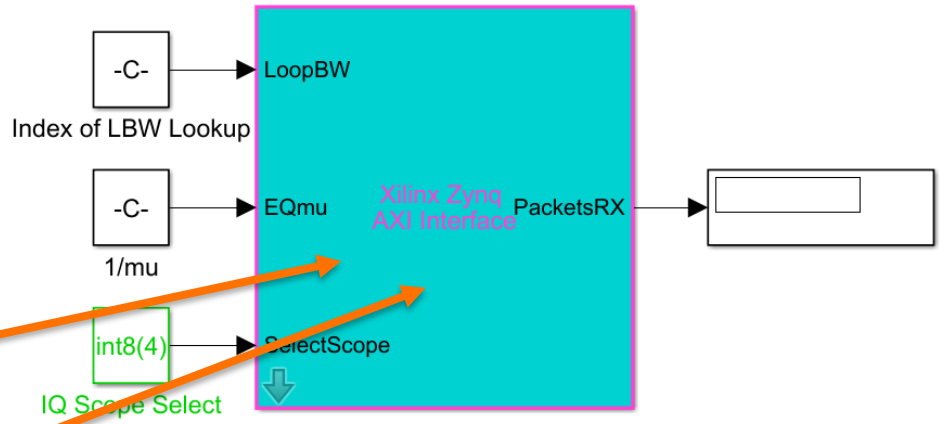
- Three Step Process
  - Define extra parameters as AXI
  - Generate Interface Model
  - Select External Mode



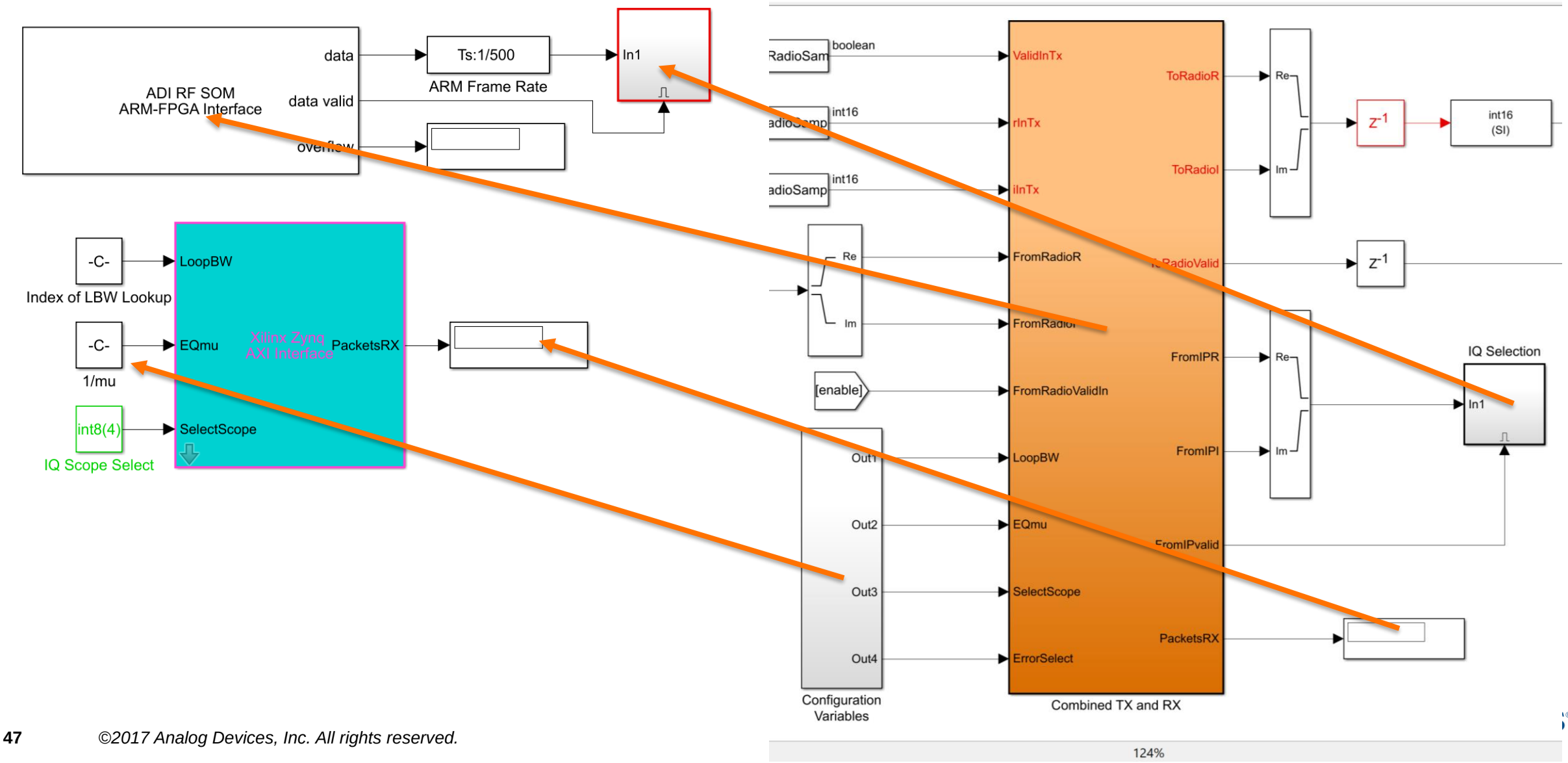
Top Level Model API For External Mode



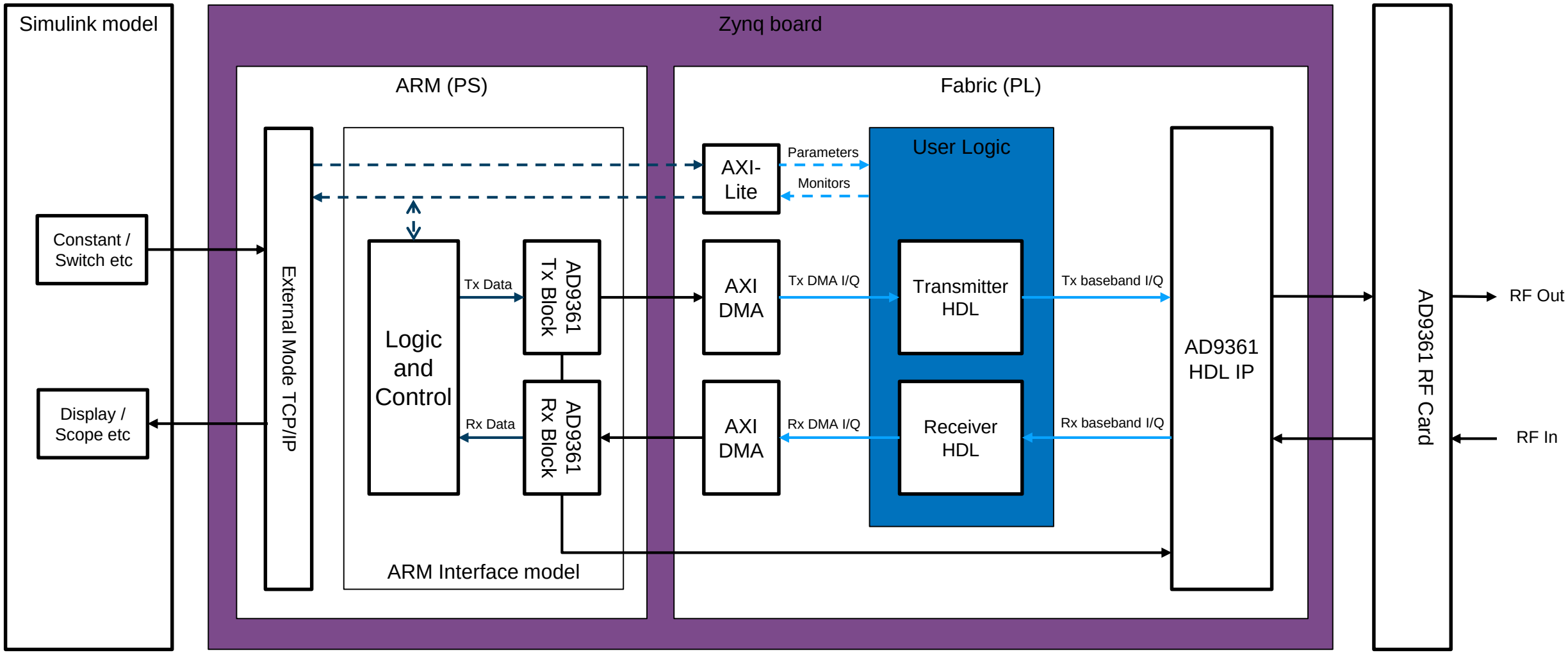
- Simulink Interface Model from *Embedded Coder*



Back In Simulink



Detailed Data Flow: Additional Control



ANALOG DEVICES
AHEAD OF WHAT'S POSSIBLE™

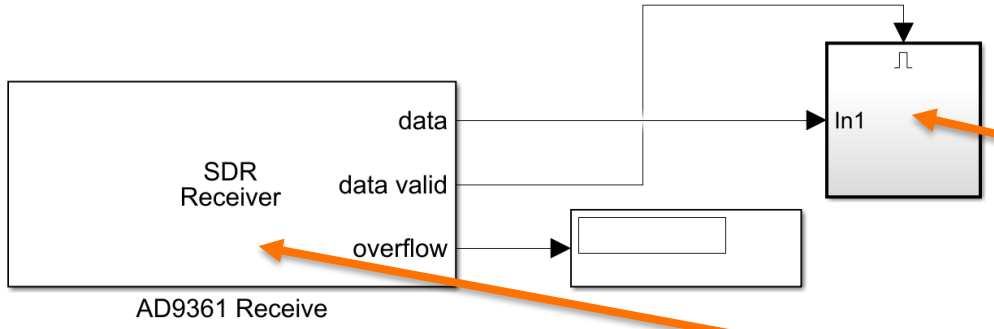
MathWorks®

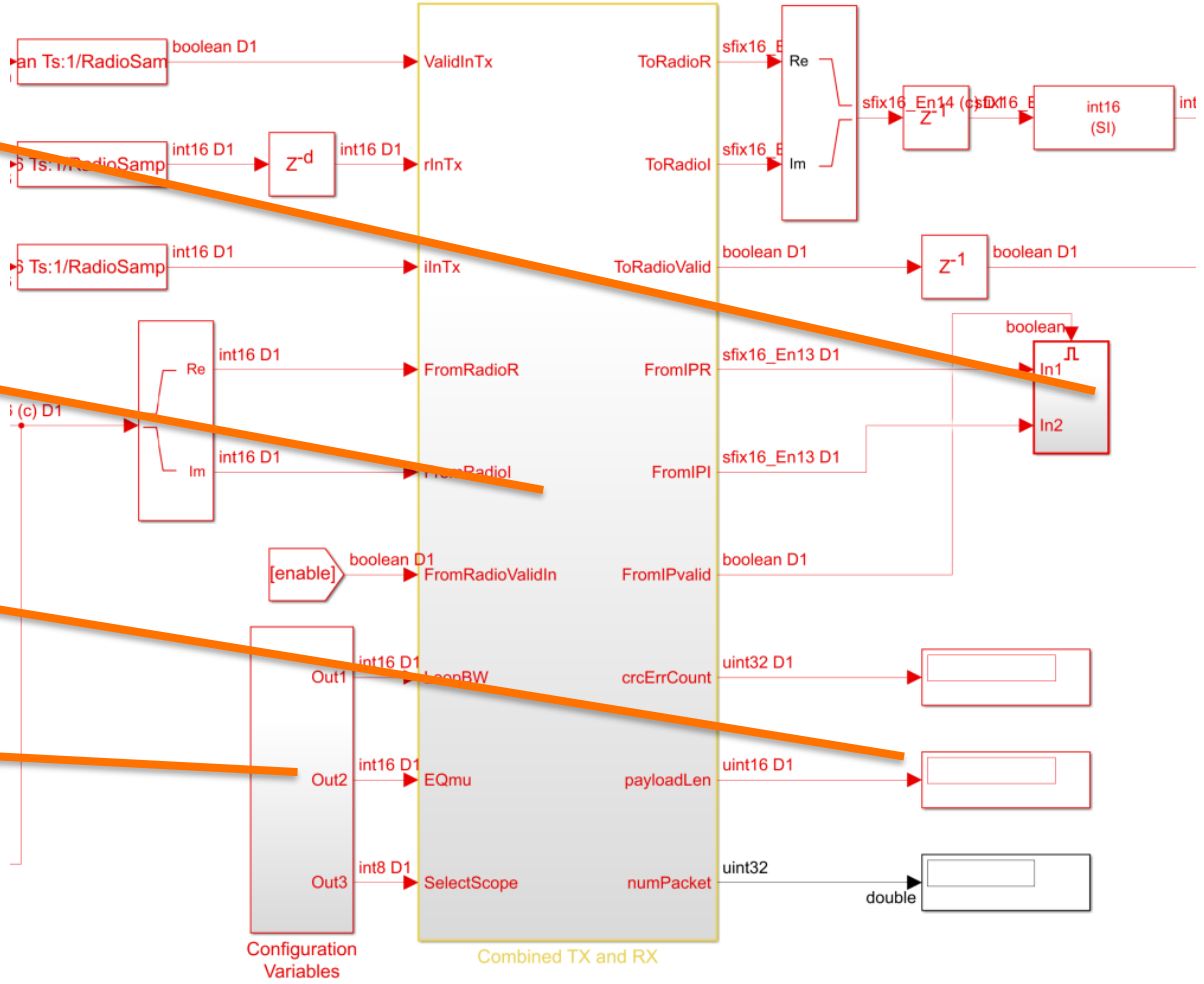
ANALOG DEVICES
AHEAD OF WHAT'S POSSIBLE™

MathWorks®

Lab 3 Part B: External Mode

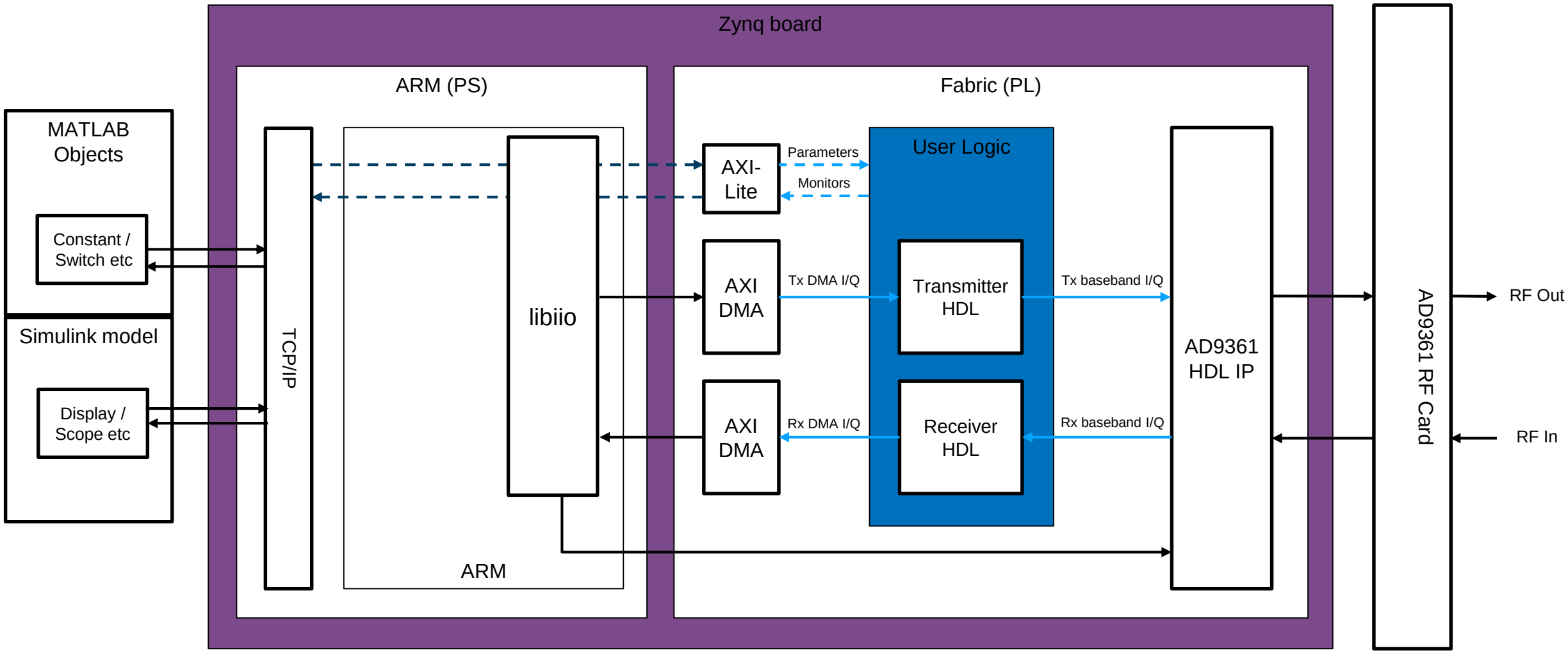
IIO AXI-MM






```
%% Writers
% Frequency Recovery Loop Bandwidth
w1 = matlabshared.libiio.aximm.write('uri',radioIP);
w1.AddressOffset = hex2dec('100');
w1.HardwareDataType='int16';
% Equalizer Step Size
w2 = matlabshared.libiio.aximm.write('uri',radioIP);
w2.AddressOffset = hex2dec('104');
w2.HardwareDataType='int16';
% IQ Scope Selection
w3 = matlabshared.libiio.aximm.write('uri',radioIP);
w3.AddressOffset = hex2dec('108');
w3.HardwareDataType='int8';
% Debug Status Signal Selection
w4 = matlabshared.libiio.aximm.write('uri',radioIP);
w4.AddressOffset = hex2dec('10C');
w4.HardwareDataType='int8';
```

Detailed Data Flow: IIO AXI-MM



ANALOG DEVICES
AHEAD OF WHAT'S POSSIBLE™

MathWorks®

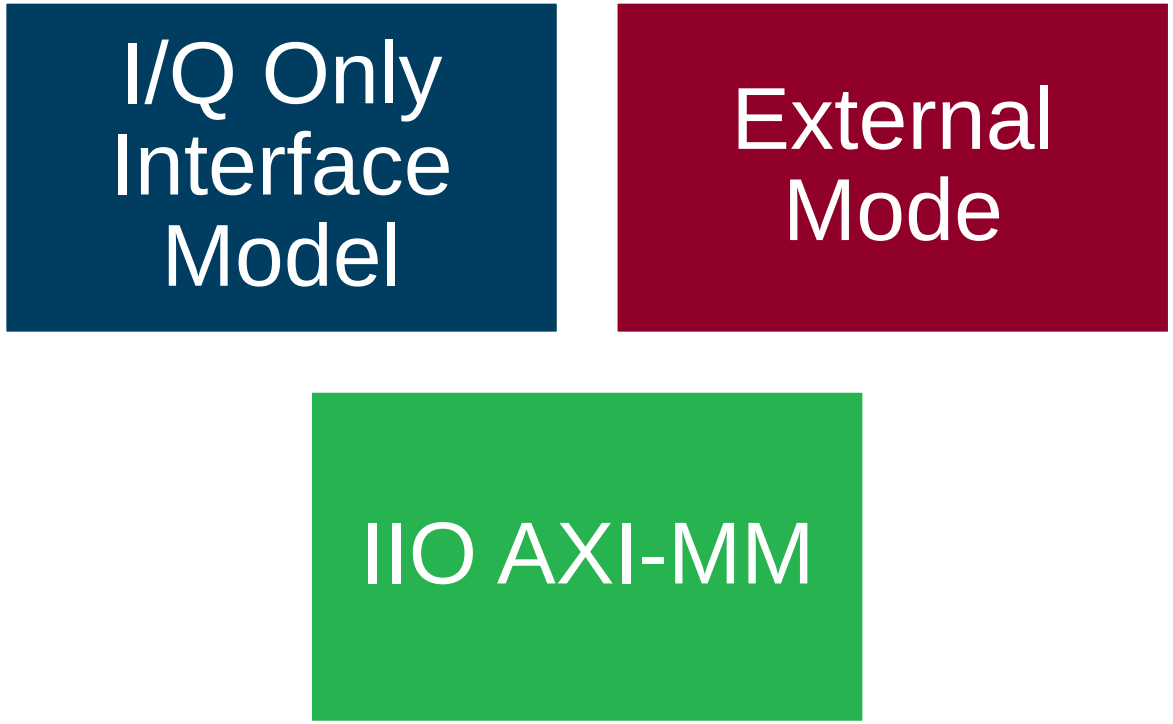
ANALOG DEVICES
AHEAD OF WHAT'S POSSIBLE™

MathWorks®

Lab 3 Part C: AXI-MM

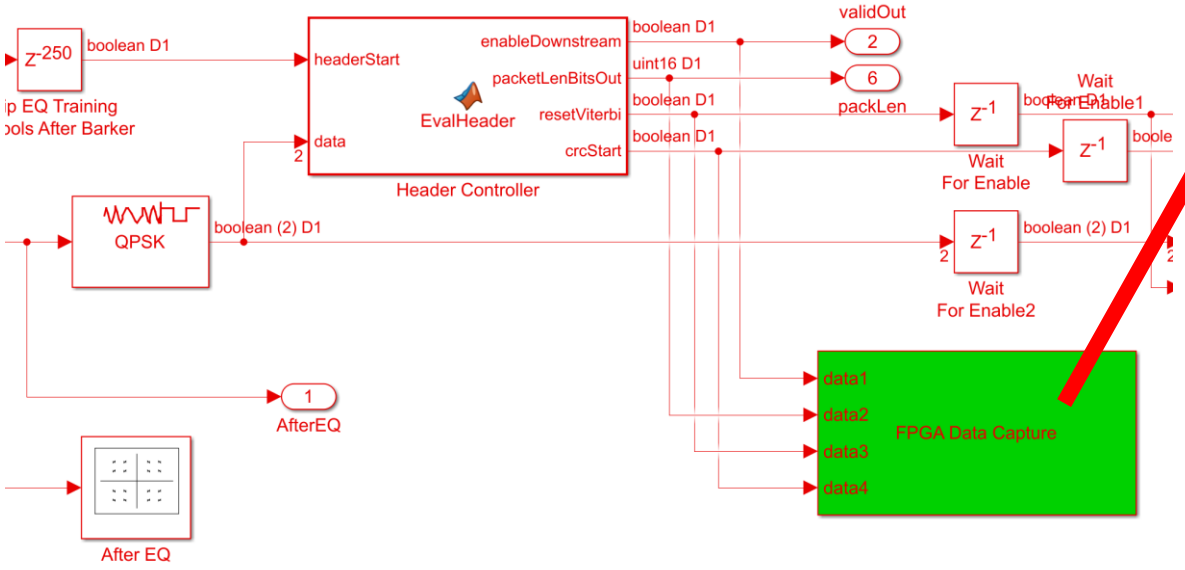
Expanding the Design Interface

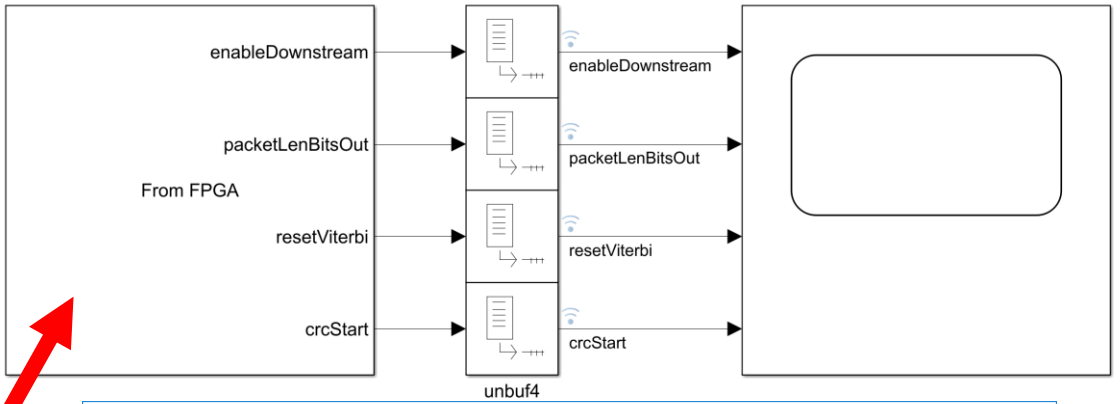
- Using External Mode can greatly increase runtime visibility
  - Additional AXI-lite interfaces for use with External Mode is not meant for high speed I/O
  - Embedded Coder becomes a requirement for External Mode
- IIO AXI-MM requires manual register assignments back in MATLAB/Simulink
- Development advantage
  - MATLAB does the plumbing work for AXI interfaces for you

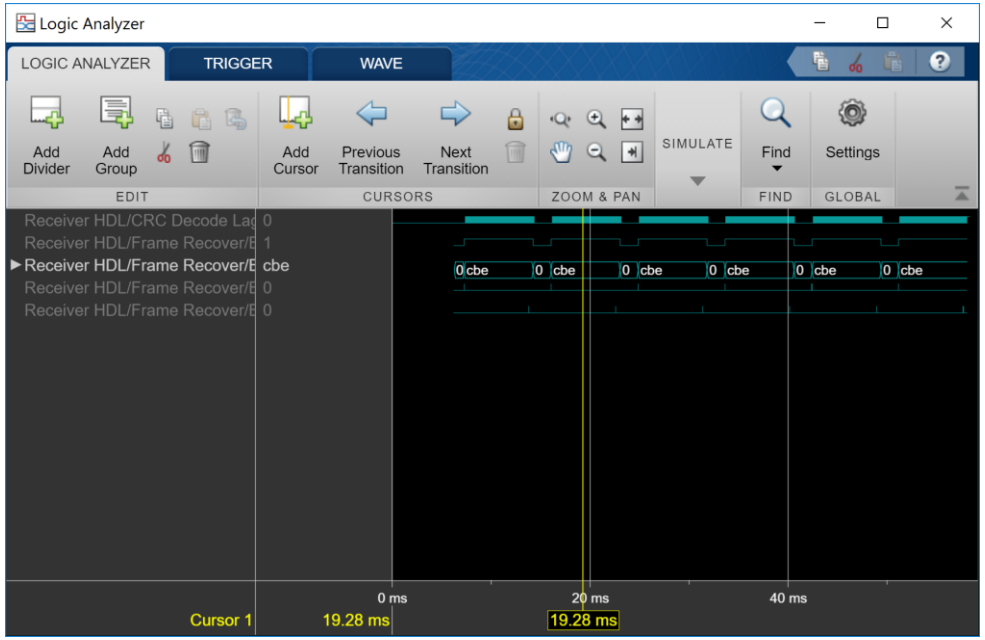


HDL Verifier: FPGA Capture Tool

- Simulink has an ILA-like option called “FPGA Capture”
- Workflow:
  - Insert block
  - After HDL generation look open new “FPGADataCapture_model.slx”
- Can display in Logic Analyzer, Simulink, or MATLAB

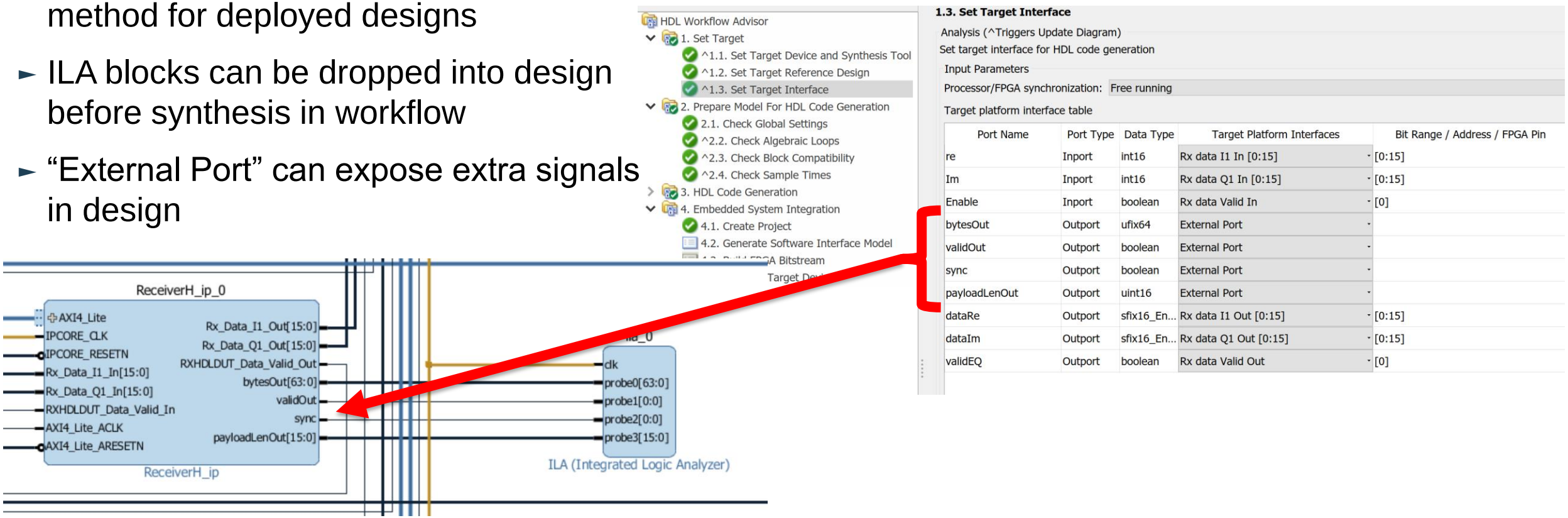






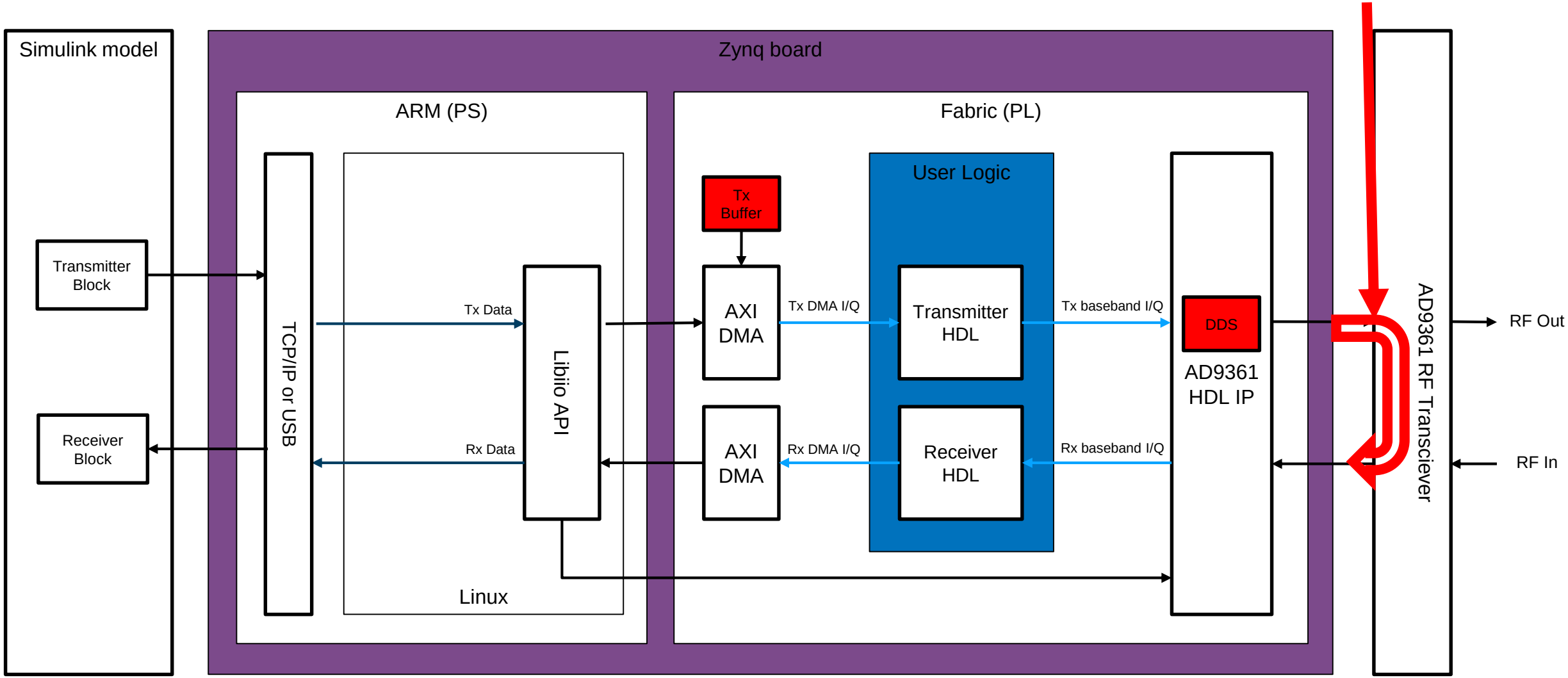
ILA Debugging Workflow

- ILA is the most common Xilinx debugging method for deployed designs
- ILA blocks can be dropped into design before synthesis in workflow
- “External Port” can expose extra signals in design



Using Hardware Features For Debug

AD9361 Digital Loopback



ANALOG DEVICES
AHEAD OF WHAT'S POSSIBLE™

MathWorks®

Deployed Debugging Options

- I/Q streaming lines in base reference design:
  - These will always exists and always must be mapped
  - Can handle high speed data
- External Mode:
  - Utilize for low-speed analysis and tuning
- IIO AXI-MM
  - Requires manual mapping
  - Works at high speed without Embedded Coder
- FPGA Capture:
  - Timing diagram debugging
  - Very useful for debugging IP integrations
- Vivado ILA (Chip Scope)
  - Timing diagram debugging

ANALOG DEVICES
AHEAD OF WHAT'S POSSIBLE™

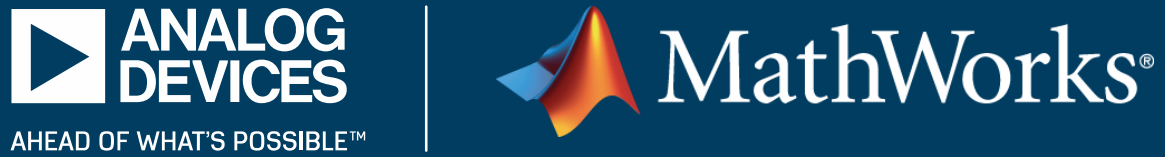
MathWorks®

Morning Review and Questions

Morning Review

- Design is likely to change when moving to HDL capable design
- Many tools and features to help get to HDL generation
- There are many options for debugging a deployed design
  - AXI-MM
  - External Mode
  - Reuse of Reference IQ Interfaces
  - FPGA Capture
  - Xilinx ILA

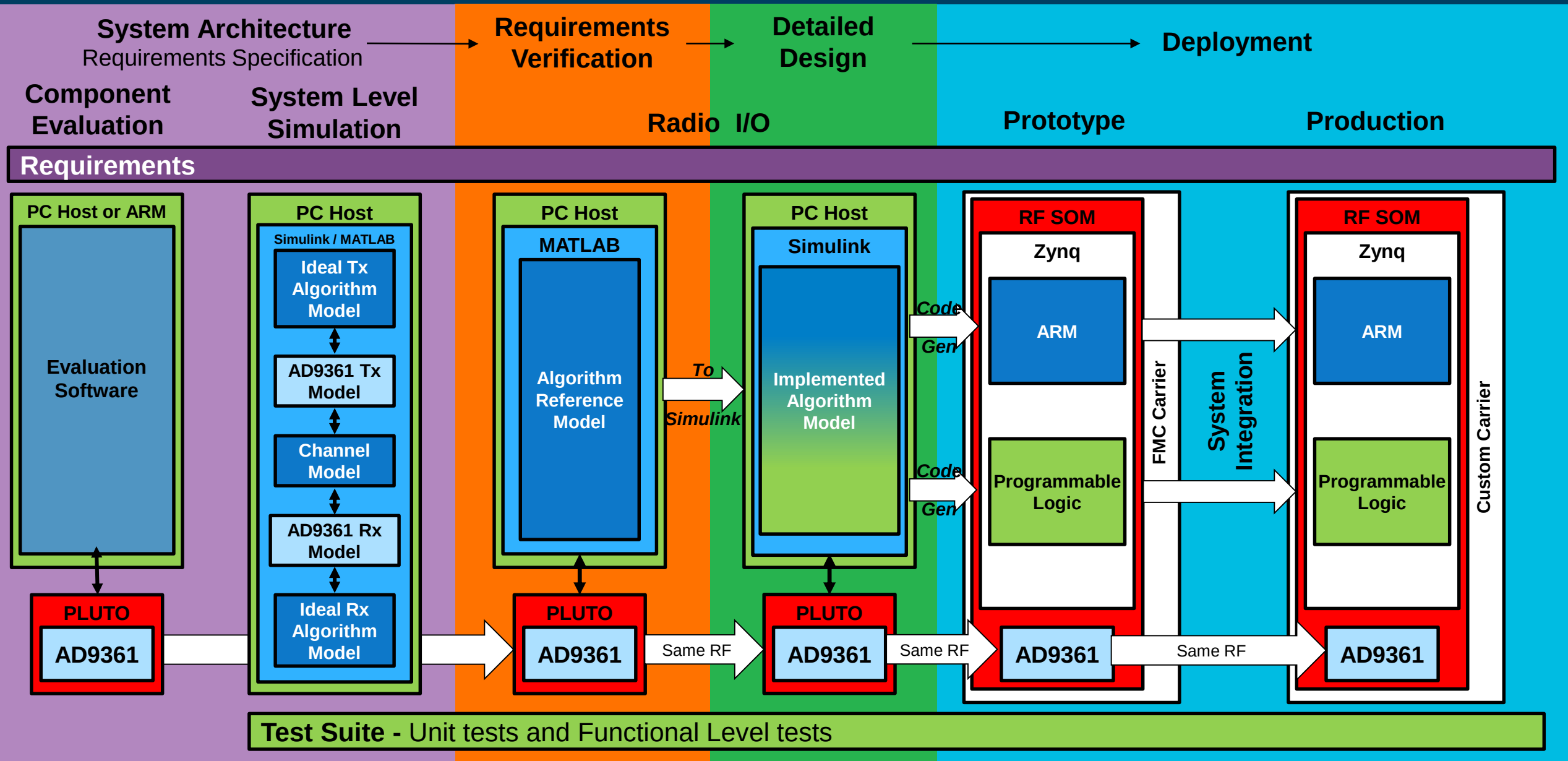
Lunch



Moving To Production

TRAVIS COLLINS, PHD

SDR Workflow Review



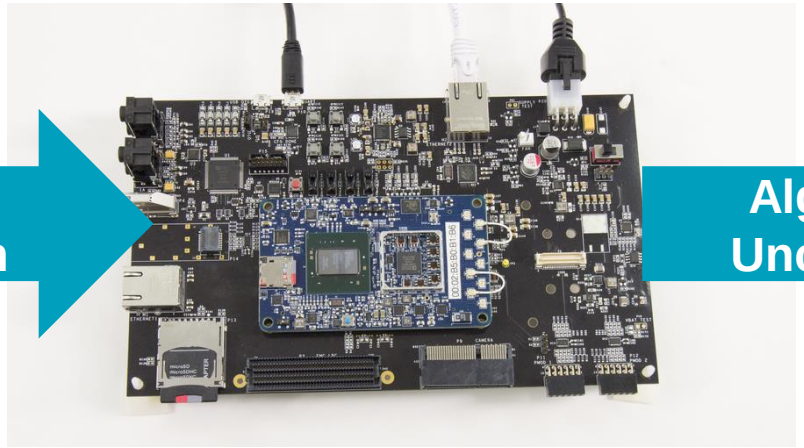
Hardware Evolution: Concept To Production

- Algorithm development and test data capture
- Inexpensive and same transceiver family
- Ideal for desktop/laptop prototyping
- AD936X transceiver

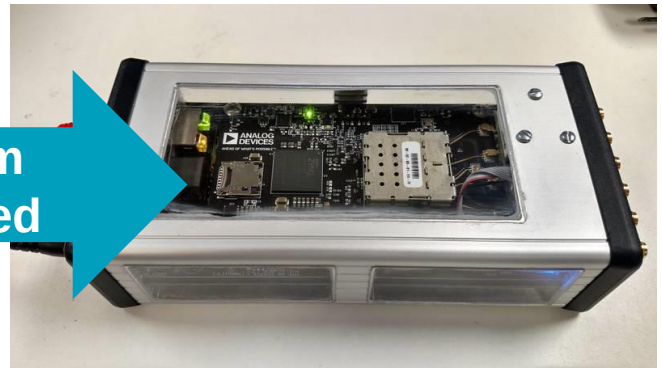
- HDL focused prototype development
- Standard reference design with significant IO
- Same RF transceiver

- Specialized hardware with custom device tree
- Same RF transceiver

Algorithm Conversion



Algorithm Unchanged



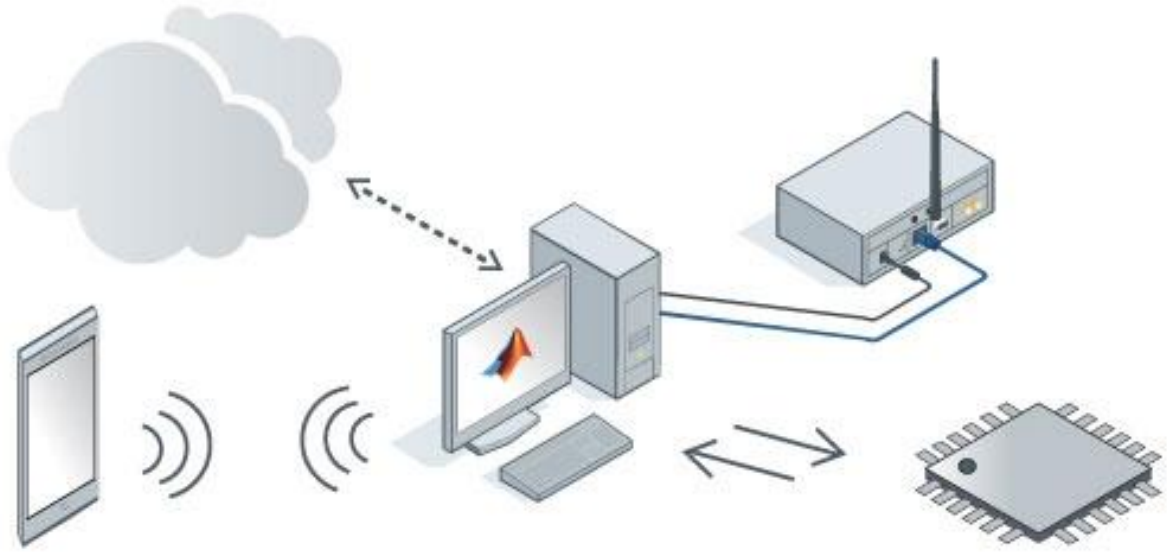
ANALOG DEVICES
AHEAD OF WHAT'S POSSIBLE™

MathWorks®

Connecting MATLAB and Simulink to Hardware

- MATLAB and Simulink are used for algorithm development and code generation
  - Not tied to any specific hardware platform

- Need a bridge to hardware system deployment
  - Hardware Support Package (HSP)
  - Reference Designs for custom hardware

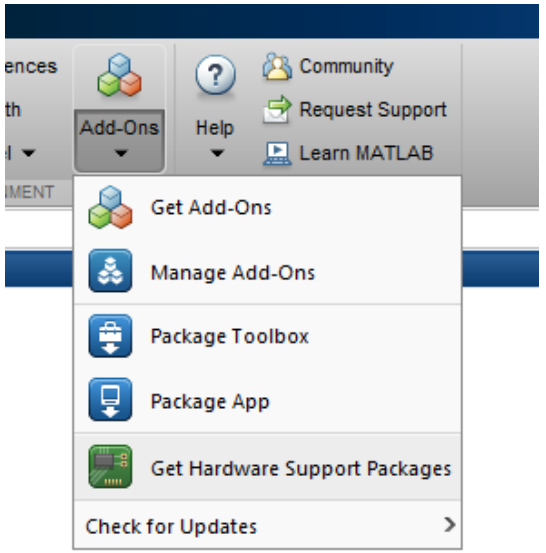


ANALOG DEVICES
AHEAD OF WHAT'S POSSIBLE™

MathWorks®

Support Packages

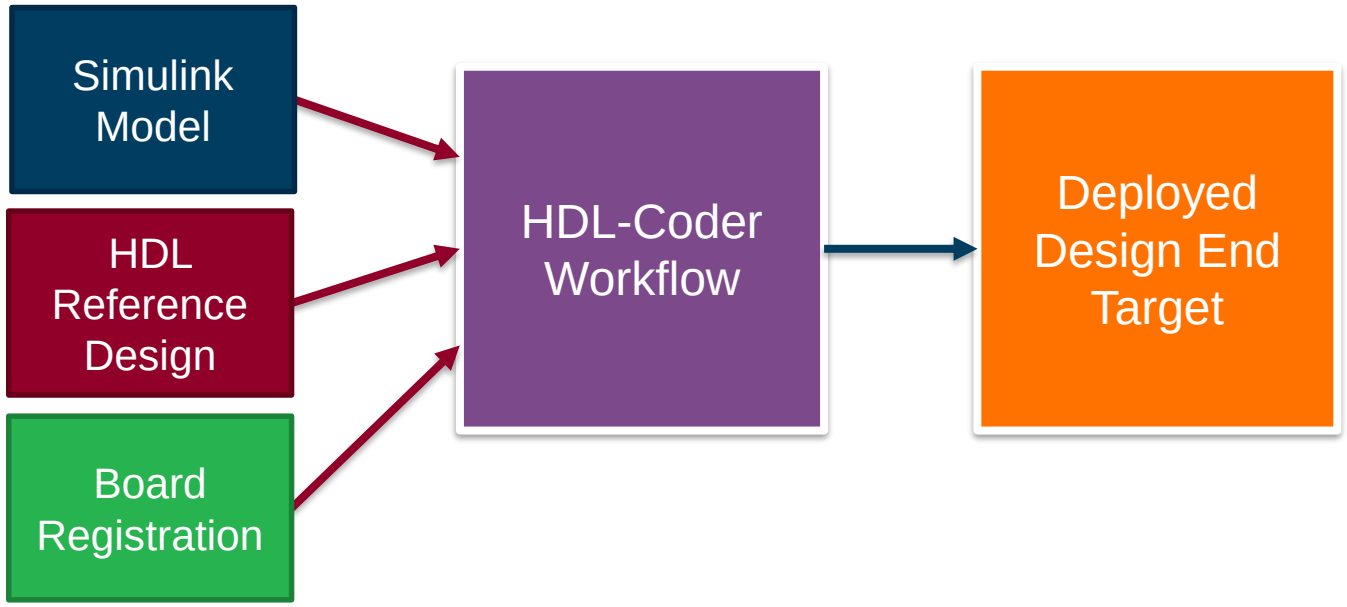
- The bridge between MATLAB and Simulink and Hardware
- Enable radio I/O, prototyping, and production deployment
- **Hardware Support Packages** are available via the add-on explorer in MATLAB
  - Provide board-specific **reference designs** for C and HDL code generation
  - Provide portable Linux drivers for data I/O
- Third-party-authored **reference designs** enable custom hardware targeting
  - Leverages published APIs

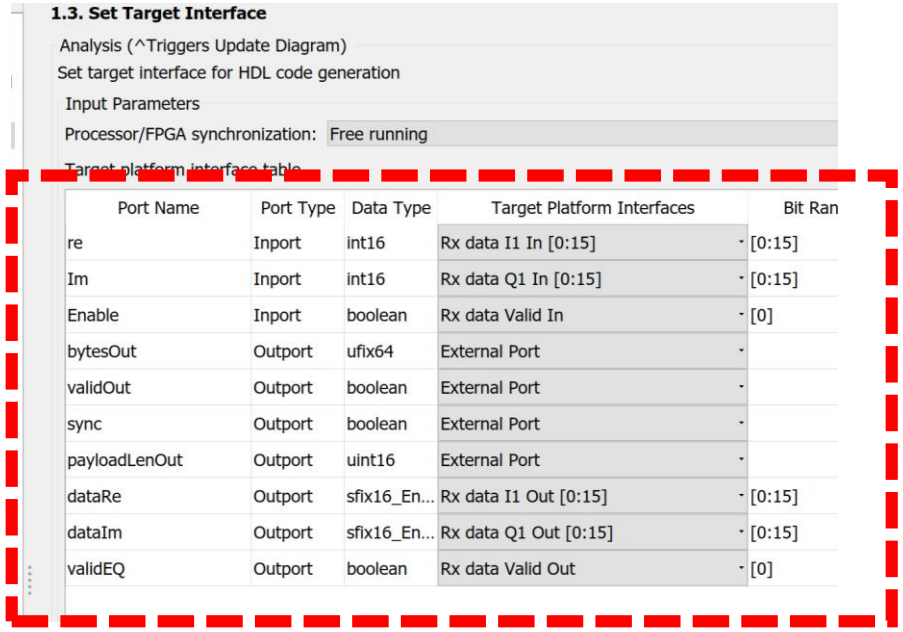


ANALOG DEVICES
AHEAD OF WHAT'S POSSIBLE™

MathWorks®

What does MathWorks mean when they refer to Board Support Package (BSP)





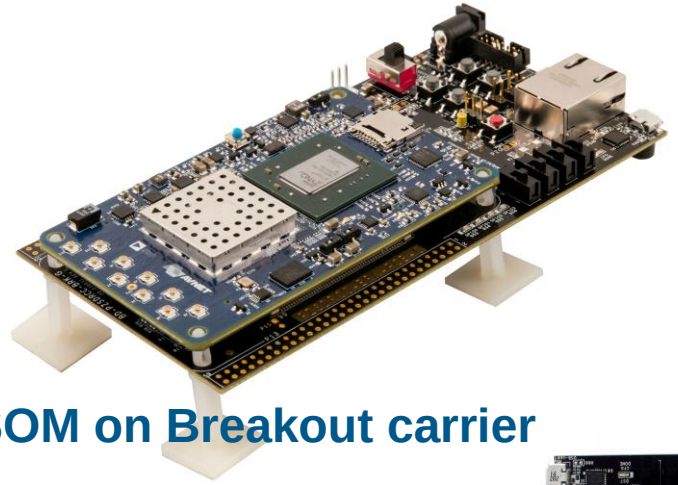
Port Name	Port Type	Data Type	Target Platform Interfaces	Bit Ran
re	Inport	int16	Rx data I1 In [0:15]	[0:15]
Im	Inport	int16	Rx data Q1 In [0:15]	[0:15]
Enable	Inport	boolean	Rx data Valid In	[0]
bytesOut	Outport	ufix64	External Port	
validOut	Outport	boolean	External Port	
sync	Outport	boolean	External Port	
payloadLenOut	Outport	uint16	External Port	
dataRe	Outport	sfix16_En...	Rx data I1 Out [0:15]	[0:15]
dataIm	Outport	sfix16_En...	Rx data Q1 Out [0:15]	[0:15]
validEQ	Outport	boolean	Rx data Valid Out	[0]

- Hardware Support Package (HSP)
  - Standard development kits
  - Fixed reference designs
  - Fixed board registration
  - End targets:
    - MATLAB/Simulink
    - ARM codegen application

- Board Support Package (BSP)
  - Custom boards (ex: PackRF board)
  - Custom reference designs
  - Custom registration API
  - End target:
    - Up to user
    - TUN/TAP in our design

Custom Carrier Boards - Motivation for BSP and Deployed Designs

- **Each application has specific requirements in terms of peripherals and interfaces**
- **Customers design their own application specific carrier boards**
  - SOM connectors
  - SOM power
- **Custom add-on boards**
  - Rx & Tx filters
  - Tx power, Rx sensitivity increase
  - Application specific RF control - TDD
- **Custom hardware will require changes**
  - HDL reference design
  - Software infrastructure



RFSOM on Breakout carrier

RFSOM bottom

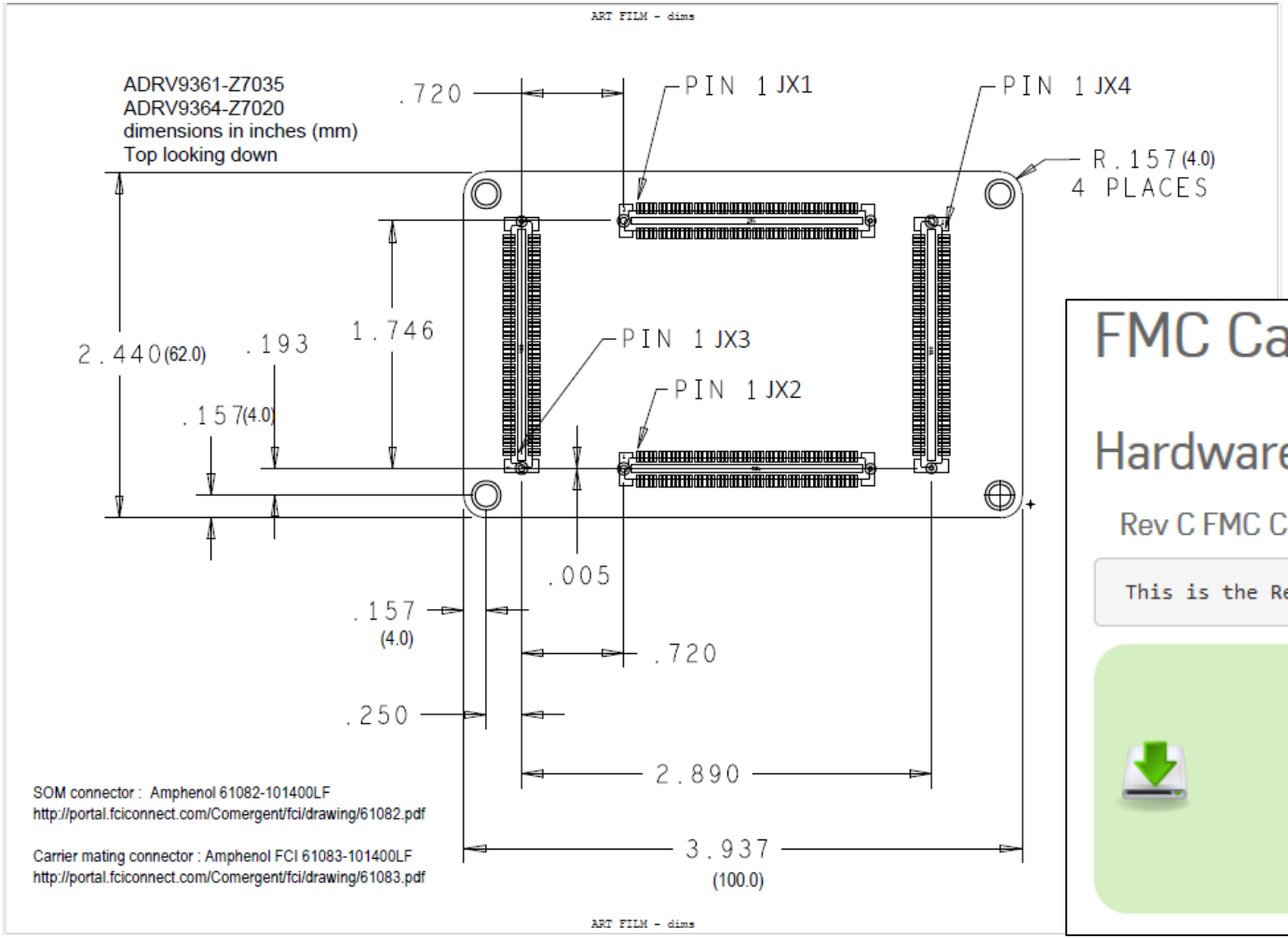
Breakout carrier top

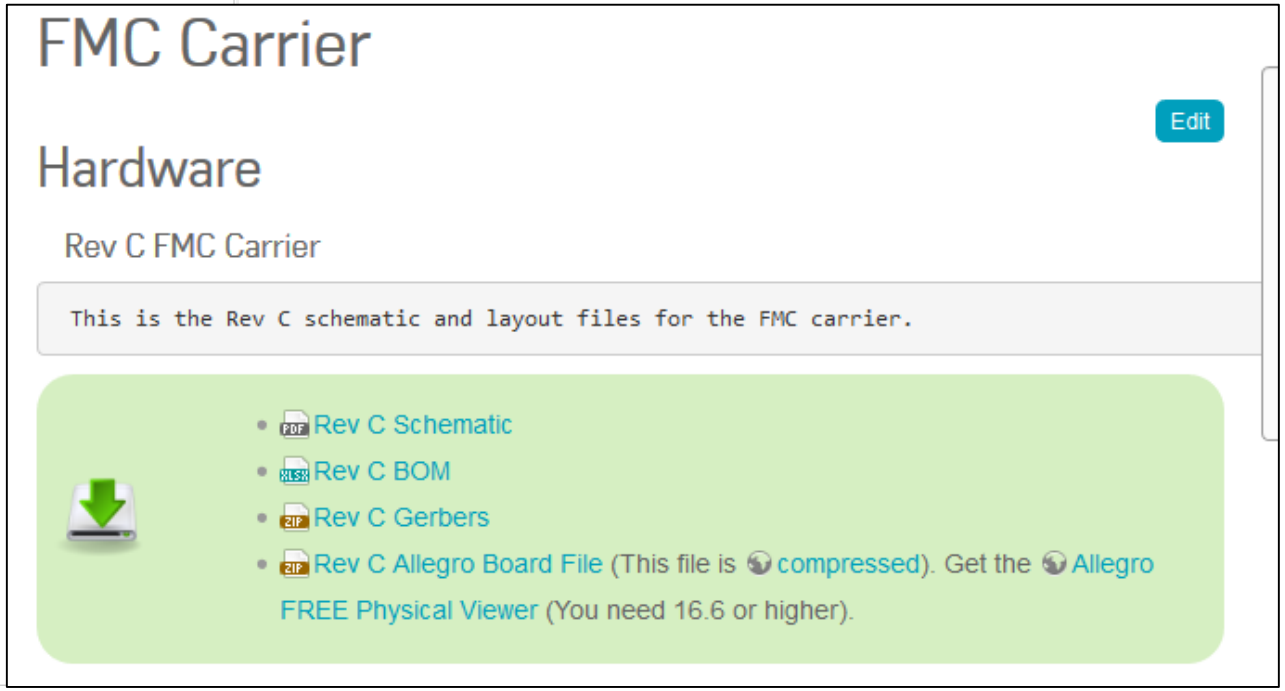
TDD personality card

ANALOG DEVICES
AHEAD OF WHAT'S POSSIBLE™

MathWorks®

Build Your Own

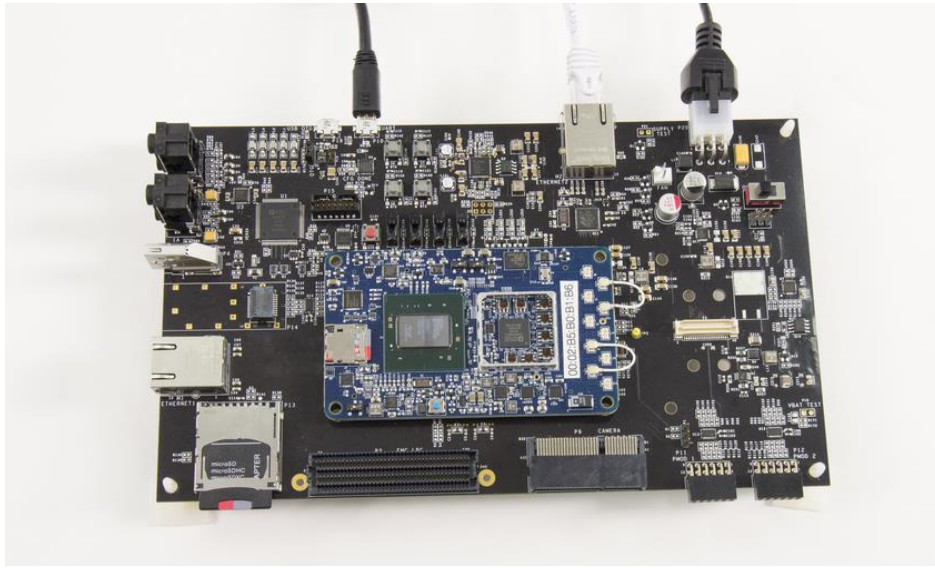




ANALOG DEVICES
AHEAD OF WHAT'S POSSIBLE™

MathWorks®

Moving to the Production Design



- RFSOM on FMC Carrier Board

Algorithm Unchanged

Same RF-SOM

New Carrier Board

- RFSOM on Custom Carrier Board

Modified Reference Design

Modified Linux Kernel

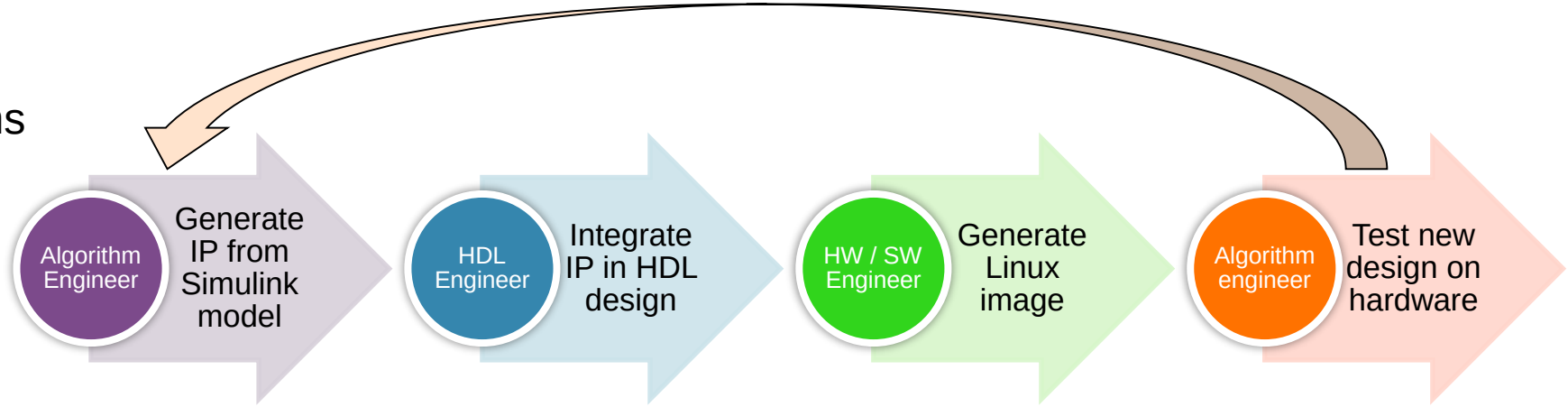
Provided by ADI For PackRF In BSP

AHEAD OF WHAT'S POSSIBLE™

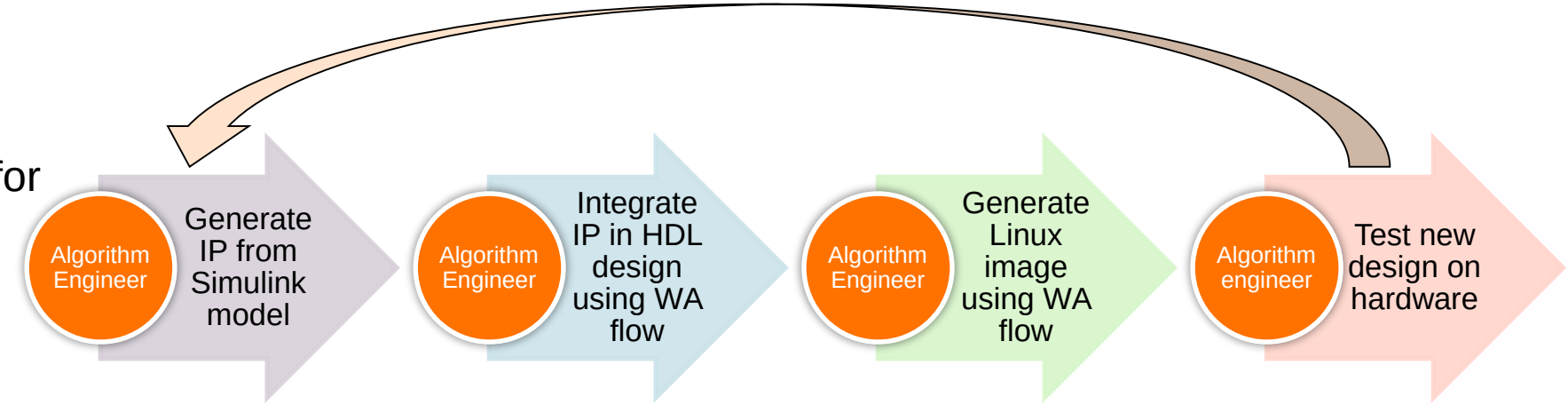
MathWorks®

Development flows on custom hardware

- **Without custom BSP**
  - Back & forth between teams
  - Interface definition mismatch probability
  - Prone to errors between steps

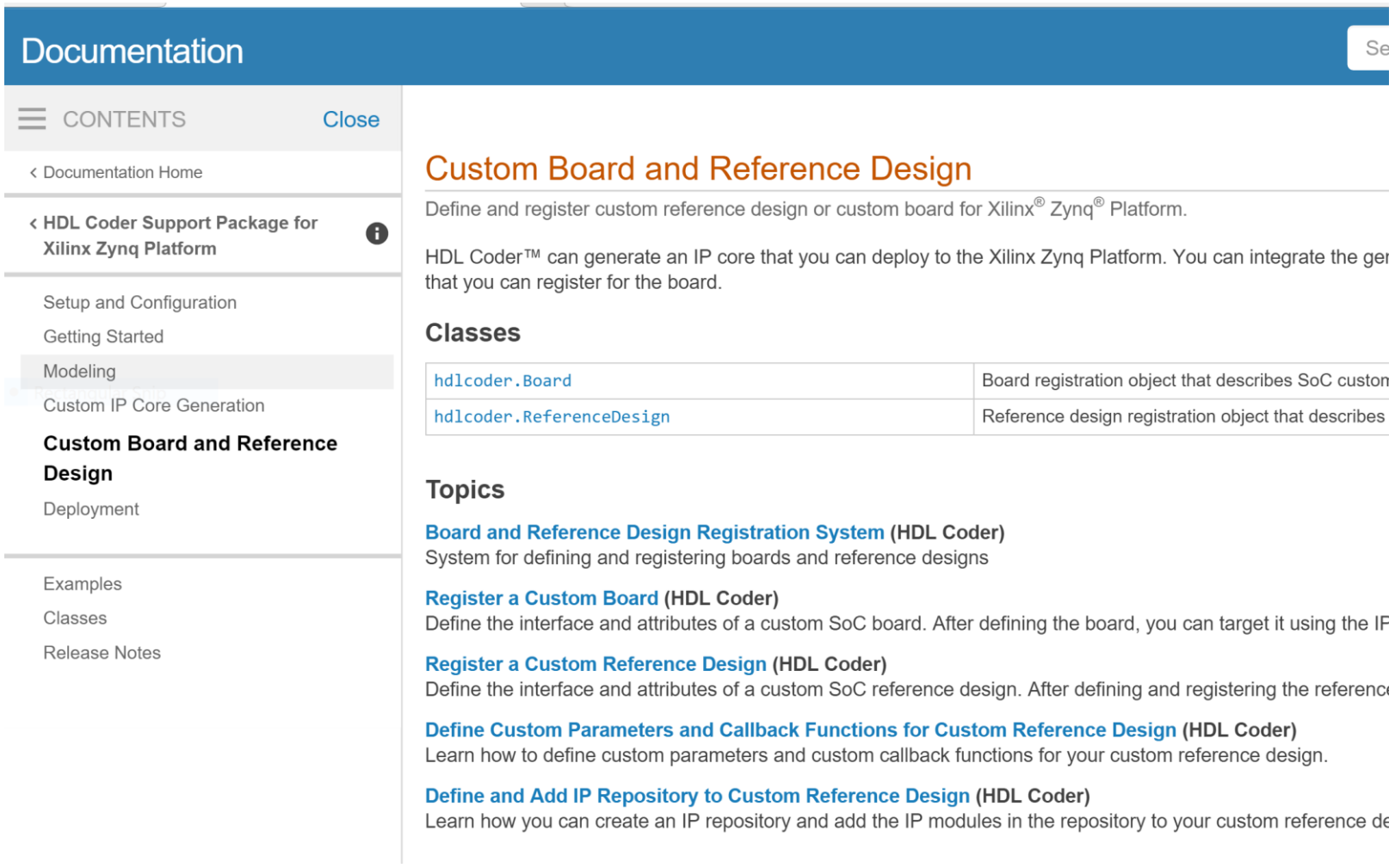


- **With custom BSP**
  - Same person does all the steps
  - Stay in the same flow as for the previous designs
  - Reduced integration time



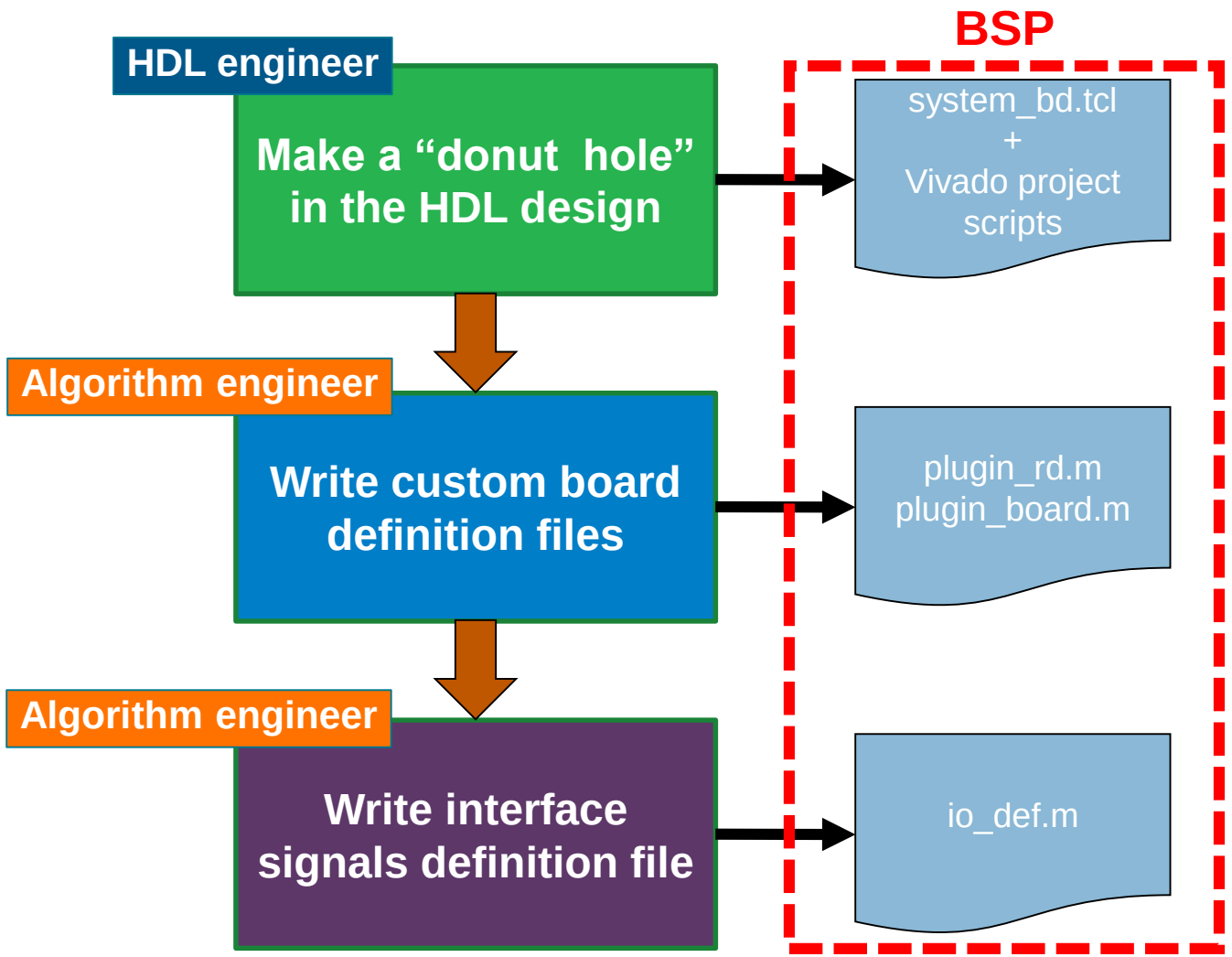
Registering a Custom Board and Reference Design (BSP)

- Process is well documented with MATLAB
- ADI BSP is an example of using this workflow
- HSPs from MathWorks are also examples of this workflow



Creating a BSP

- **What do you need to have?**
  - Custom Vivado HDL design
  - List of signals that the custom IP will connect to in the HDL design
- **What do you need to change?**
  - Make a “donut hole” in the Vivado design exposing the interface signals to the custom IP
  - Update the devicetree and Linux image to reflect HDL design changes
- **What do you need to add?**
  - Create custom board definition files
  - Create interface signals definition file



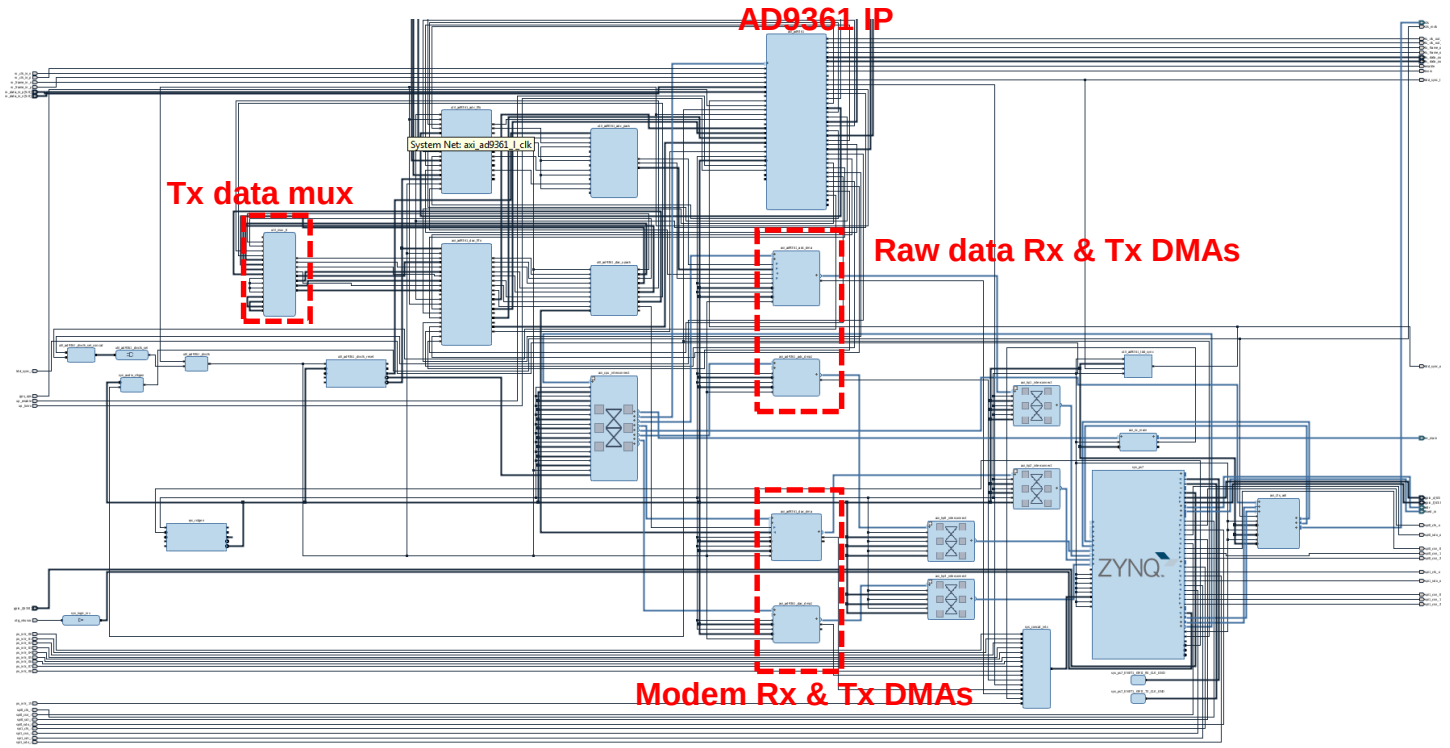
PackRF and Modem HDL Design Changes

HDL design changes goals

- Keep the Rx and Tx raw data paths intact for debugging purposes
- Provide a “donut hole” where the custom IP can be inserted

HDL design changes

- Insert a second set of Rx and Tx DMAs – the regular data paths stay intact
- Add a mux on the Tx path to select between the modem IP output and the raw data
- Add one more interface on the AXI interconnect for the modem IP
- Add two debug channels on the raw Rx data path



Interface signals

- Rx Radio I & Q – 16 bit
- Modem data from Rx DMA – 64 bit
- Data valid to Rx DMA – 1 bit
- Sync to Rx DMA – 1 bit
- Debug data – 2 x 16 bit
- Debug data ready – 1 bit
- Tx Radio I & Q – 16 bit
- Tx modem data – 64 bit
- Data valid from Tx DMA – 1 bit
- Need data Tx DMA – 1 bit
- Tx mux selection – 1 bit

ANALOG DEVICES
AHEAD OF WHAT'S POSSIBLE™

MathWorks®

Creating custom hardware support for the HDL Workflow Advisor

A practical example

- Interface signal definitions

```
function add_rx_io(hRD)

% add AXI4 and AXI4-Lite slave interfaces
hRD.addAXI4SlaveInterface( ...
    'InterfaceConnection', 'axi_cpu_interconnect/M05_AXI', ...
    'BaseAddress',         '0x43C00000', ...
    'MasterAddressSpace',  'sys_ps7/Data');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Rx Reference design interfaces
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
hRD.addInternalIOInterface( ...
    'InterfaceID',    'IP Data Valid OUT', ...
    'InterfaceType',  'OUT', ...
    'PortName',       'dut_data_valid', ...
    'PortWidth',      1, ...
    'InterfaceConnection', 'axi_ad9361_adc_dma/fifo_wr_en', ...
    'IsRequired',     false);

hRD.addInternalIOInterface( ...
    'InterfaceID',    'IP Data OUT', ...
    'InterfaceType',  'OUT', ...
    'PortName',       'dut_data_out', ...
    'PortWidth',      64, ...
    'InterfaceConnection', 'axi_ad9361_adc_dma/fifo_wr_din', ...
    'IsRequired',     false);

hRD.addInternalIOInterface( ...
    'InterfaceID',    'ADC DMA sync', ...
    'InterfaceType',  'OUT', ...
    'PortName',       'adc_dma_fifo_wr_sync', ...
    'PortWidth',      1, ...
    'InterfaceConnection', 'axi_ad9361_adc_dma/fifo_wr_sync', ...
    'IsRequired',     false);

hRD.addInternalIOInterface( ...
    'InterfaceID',    'AD9361 ADC Data I0', ...
    'InterfaceType',  'IN', ...
    'PortName',       'sys_wfifo_0_dma_wdata', ...
    'PortWidth',      16, ...
    'InterfaceConnection', 'util_ad9361_adc_fifo/dout_data_0', ...
    'IsRequired',     false);

hRD.addInternalIOInterface( ...
    'InterfaceID',    'AD9361 ADC Data Q0', ...
```

- Custom board definition files

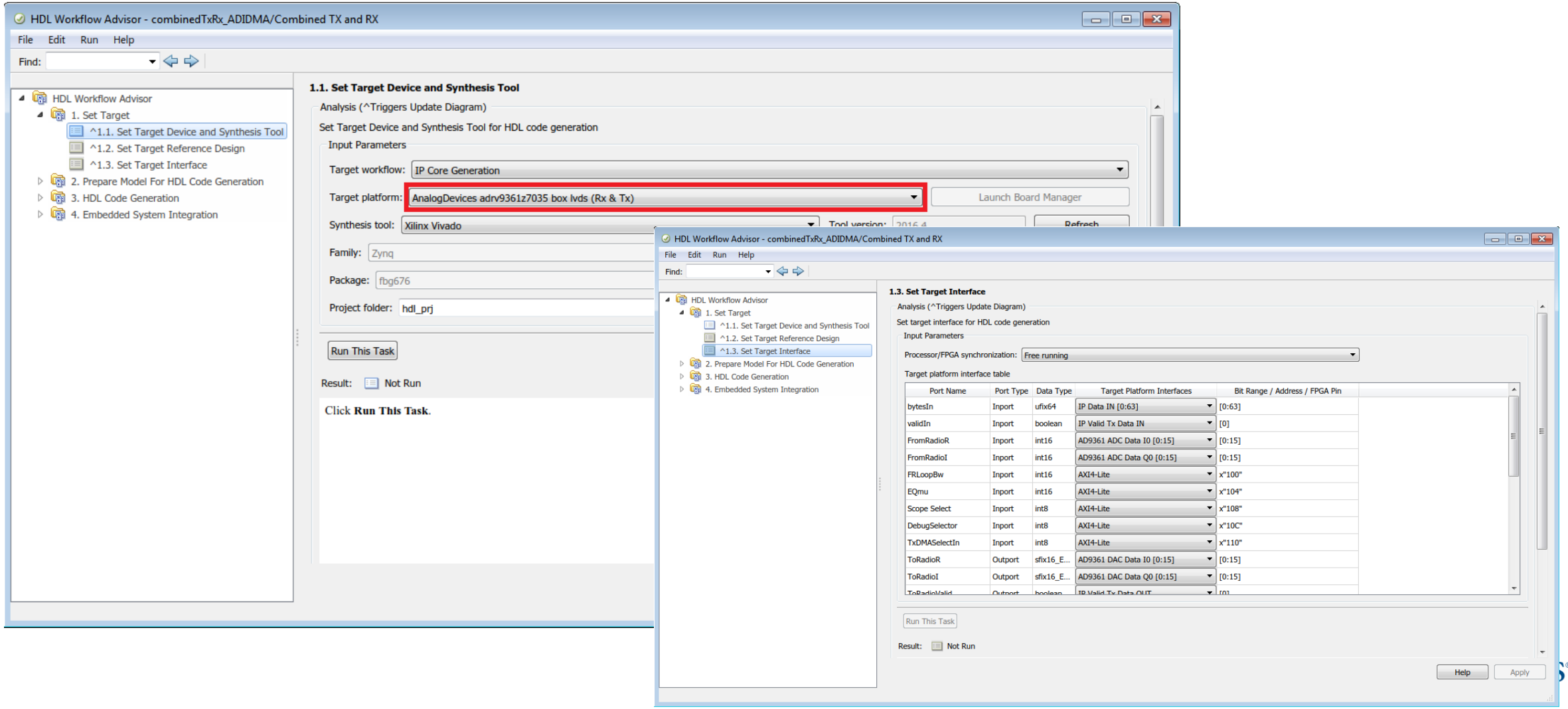
```
function [rd, boardName] = hdlcoder_ref_design_customization
% Reference design plugin registration file
% 1. The registration file with this name inside of a board plugin folder
%    will be picked up
% 2. Any registration file with this name on MATLAB path will also be picked up
% 3. The registration file returns a cell array pointing to the location of
%    the reference design plugins
% 4. The registration file also returns its associated board name
% 5. Reference design plugin must be a package folder accessible from
%    MATLAB path, and contains a reference design definition file

%   Copyright 2013-2014 The MathWorks, Inc.

rd = {'AnalogDevices.RFSOM2.ccbox_modem.rx_tx.plugin_rd', ...
      };

boardName = 'Ana

end
```

```
function hRD = plugin_rd(board, design)
% Reference design definition

%   Copyright 2014-2015 The MathWorks, Inc.

% Construct reference design object
hRD = hdlcoder.ReferenceDesign('SynthesisTool', 'Xilinx Vivado');

% Create the reference design for the SOM-only
% This is the base reference design that other RDs can build upon
hRD.ReferenceDesignName = sprintf('RFSOM2 %s Base System (Vivado 2016.2)', upper(board));

% Determine the board name based on the design
hRD.BoardName = sprintf('AnalogDevices RFSOM2 %s (%s)', upper(board), design);

% Tool information
hRD.SupportedToolVersion = {'2016.2'};

% Get the root directory
rootDir = fileparts(strtok(mfilename('fullpath'), '+'));

% Design files are shared
hRD.SharedRD = true;
hRD.SharedRDFolder = fullfile(rootDir, 'vivado');
```

Creating custom hardware support for the HDL Workflow Advisor

A practical example (continued)

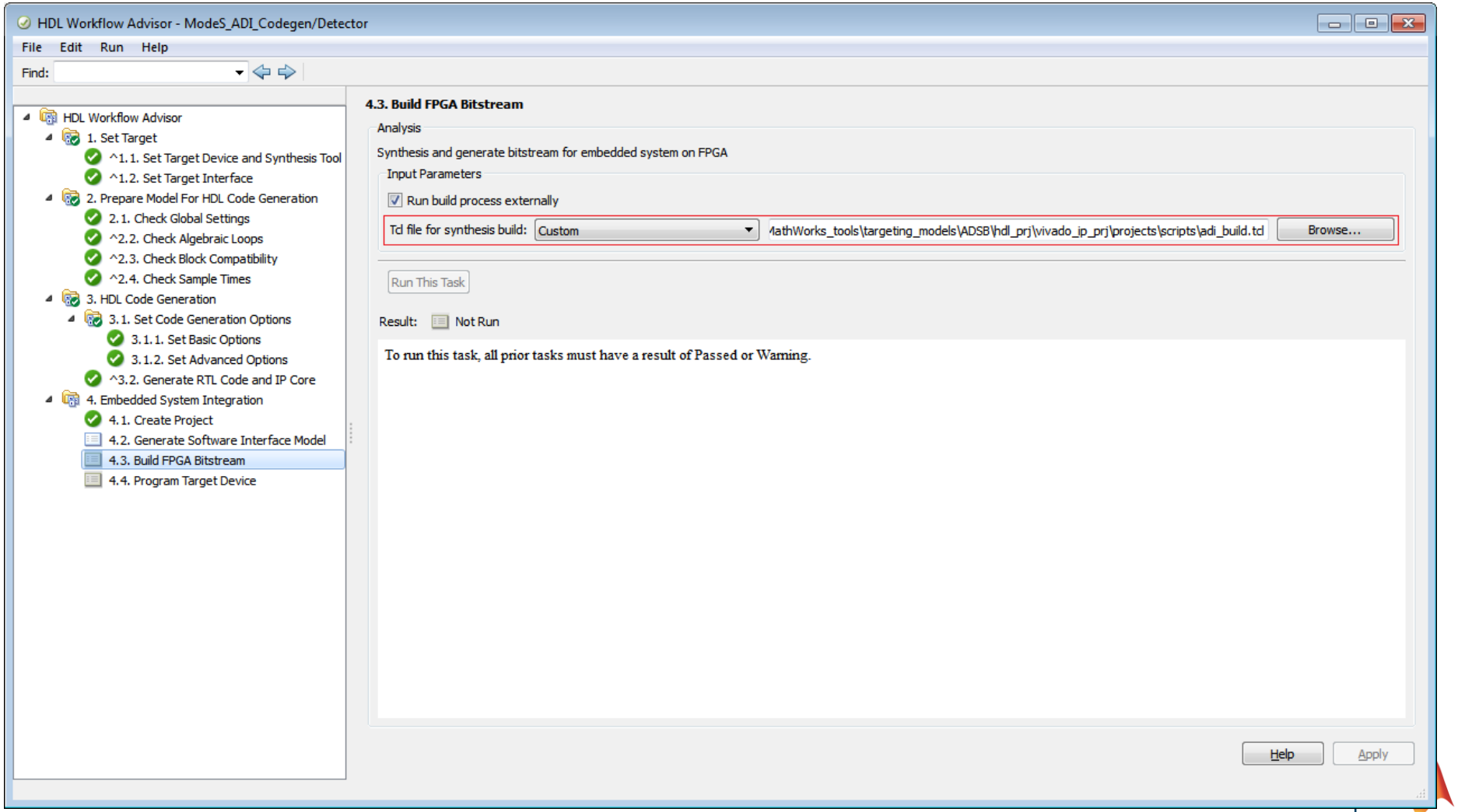
- The custom hardware support shows up in HDL Workflow Advisor



Creating custom hardware support for the HDL Workflow Advisor

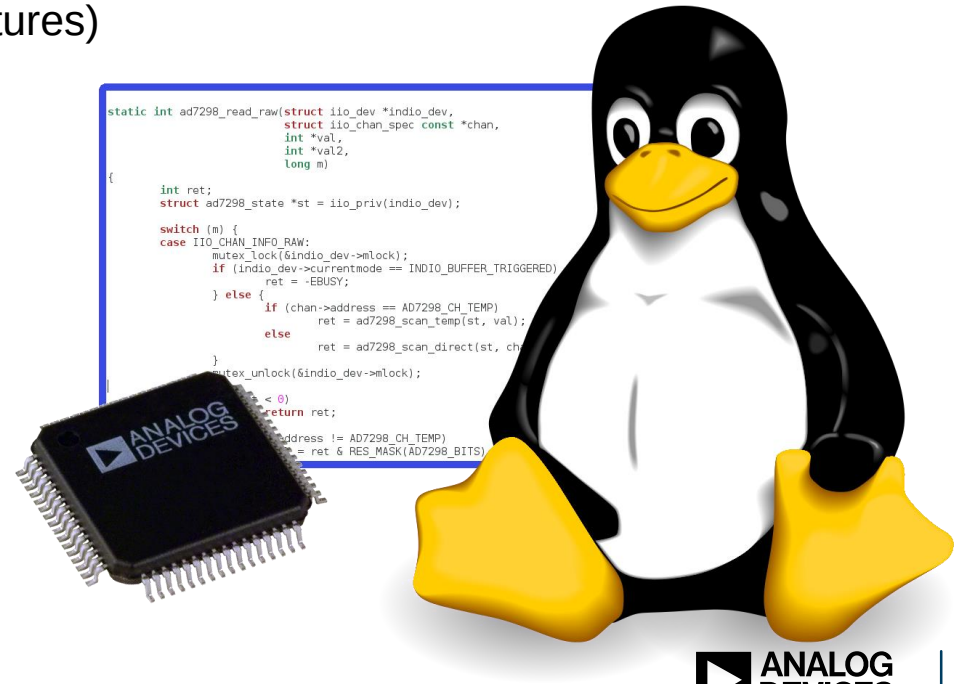
A practical example (continued)

- Run custom script for software and boot image generation



Integration with custom software – ADI Linux image

- Modern systems are diverse and complex (horizontally and vertically)
  - Different components from different vendors need to interact in the same system
  - Same component is used in different solutions
  - Linux provides interoperability and reusability
- Linux has a wide range of supported hardware
  - Excellent support for additional peripherals
    - Applications processor (30+ supported architectures)
    - Storage (SATA, SD, …)
    - Connectivity: Ethernet, Wi-Fi, USB
    - Video (HDMI, VGA, …)
    - Audio
    - Human Interface Devices
- Linux is more than the kernel
  - Large and stable software eco-system
- Total cost of ownership

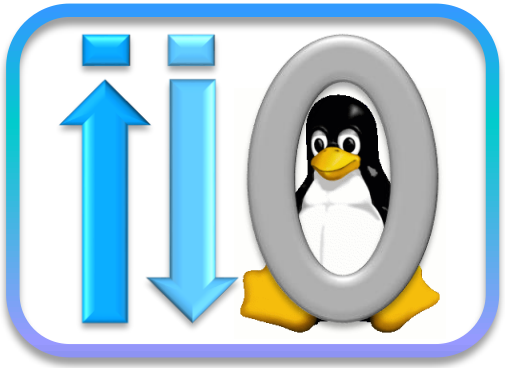


ANALOG DEVICES
AHEAD OF WHAT'S POSSIBLE™

MathWorks®

Integration with custom software – IIO, A Kernel Subsystem for Converters

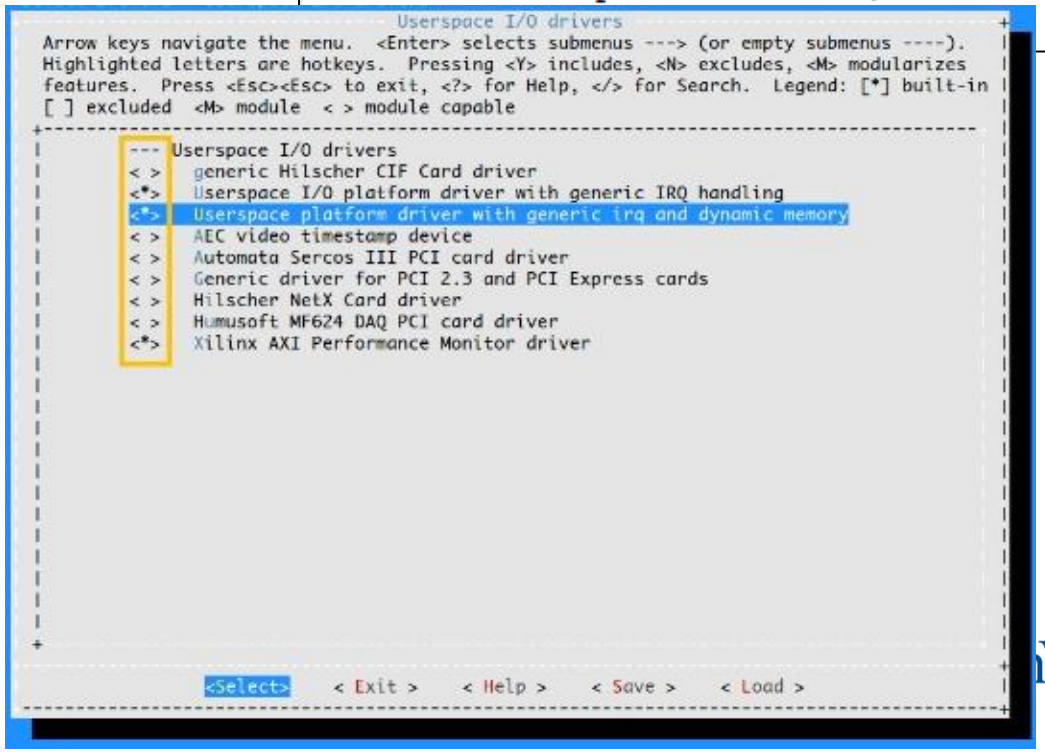
- The Linux industrial I/O subsystem is intended to provide support for devices that, in some sense, are analog-to-digital or digital-to-analog converters
  - Devices that fall into this category are:
    - Precision ADCs, high-speed ADCs
    - Precision DACs, high-speed DACs
    - Accelerometers, gyroscopes, IMUs
    - Capacitance-to-Digital converters (CDCs)
    - Pressure, proximity, temperature and light sensors
    - Health, chemical, magnetometer, amplifiers, etc.
  - Can be used on ADCs ranging from a SoC ADC to >1000 MSPS
  - Mostly focused on user-space abstraction, but also in-kernel API for other drivers exists
    - IIO to Linux input or HWMON subsystem bridges



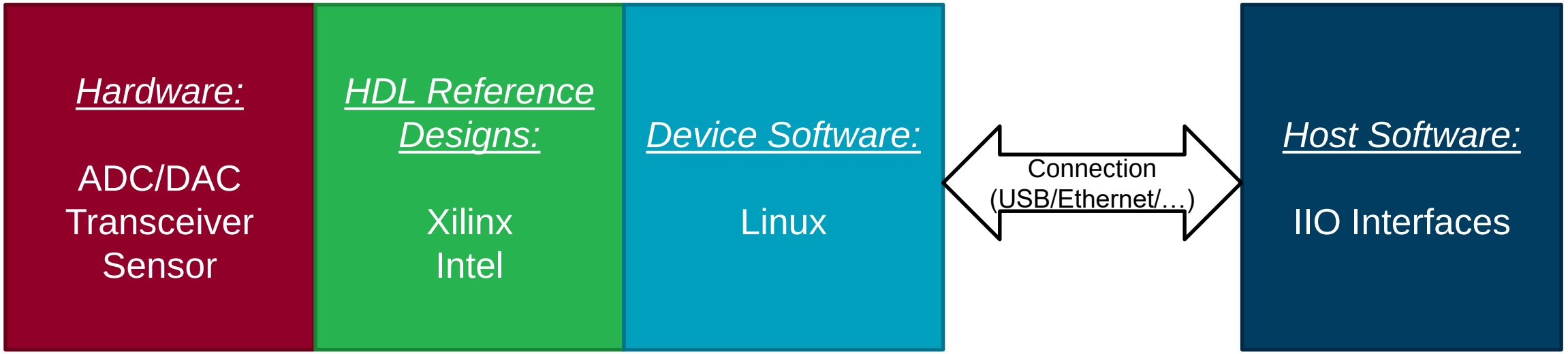
PackRF and Software Design Changes

- **HDL changes require software changes**
  - Add drivers to talk to the new IPs
  - Account for address space changes
- **Linux devicetree**
  - Add an entry to map the modem IP as uio
  - Add entries for the Rx & Tx data DMAs
  - Add entries for the custom peripherals (IMU, GPS, display, audio, …)
- **Linux image**
  - Enable the uio driver
  - Enable custom peripheral drivers

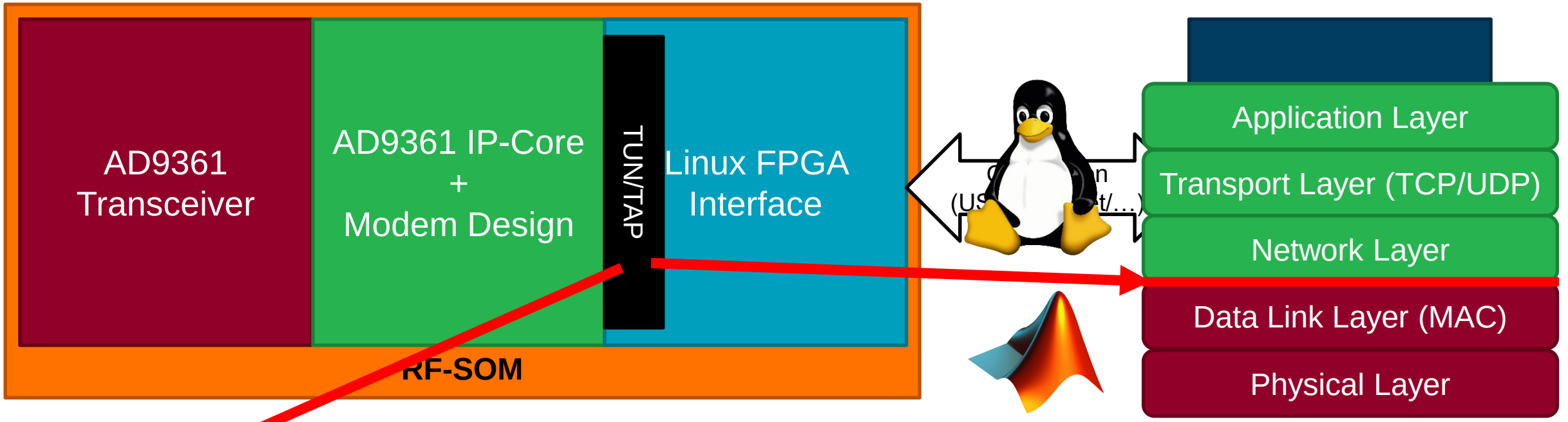
```
rx_dma@7c440000 {
        compatible = "dmem-uio";
        reg = <0x7c440000 0x10000>;
        uio,number-of-dynamic-regions = <1>;
        uio,dynamic-regions-sizes = <262144>;
};

tx_dma@7c460000 {
        compatible = "dmem-uio";
        reg = <0x7c460000 0x10000>;
        uio,number-of-dynamic-regions = <1>;
        uio,dynamic-regions-sizes = <262144>;
};

modem_ip@43C00000 {
        compatible = "dmem-uio";
        reg = <0x43C00000 0xffff>;
        interrupts = <0 55 0>;
```

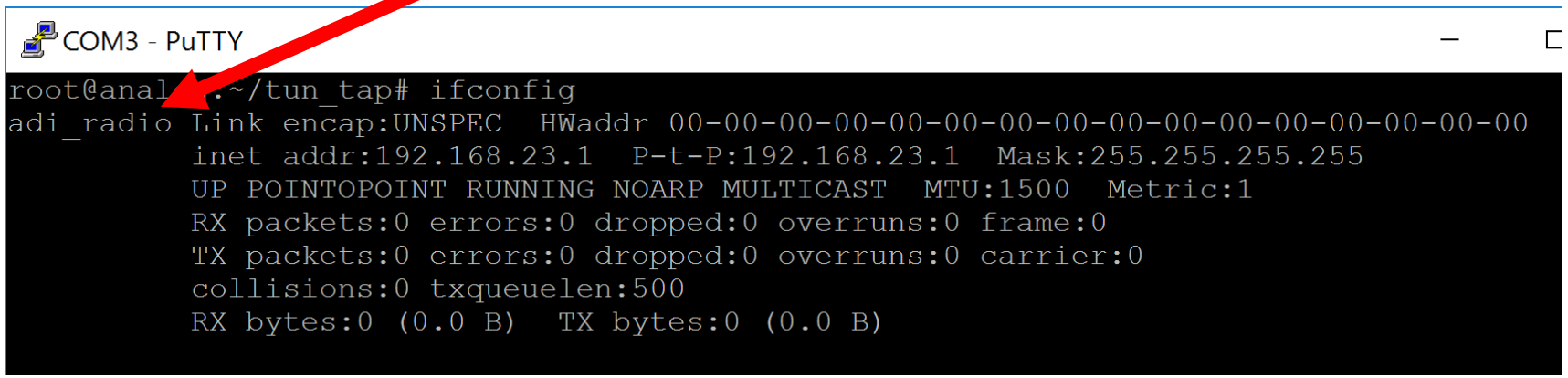


Review of MATLAB and Simulink connection



Untethering From MATLAB and Simulink





System Architecture → **Requirements Verification** → **Detailed Design** → **Deployment**

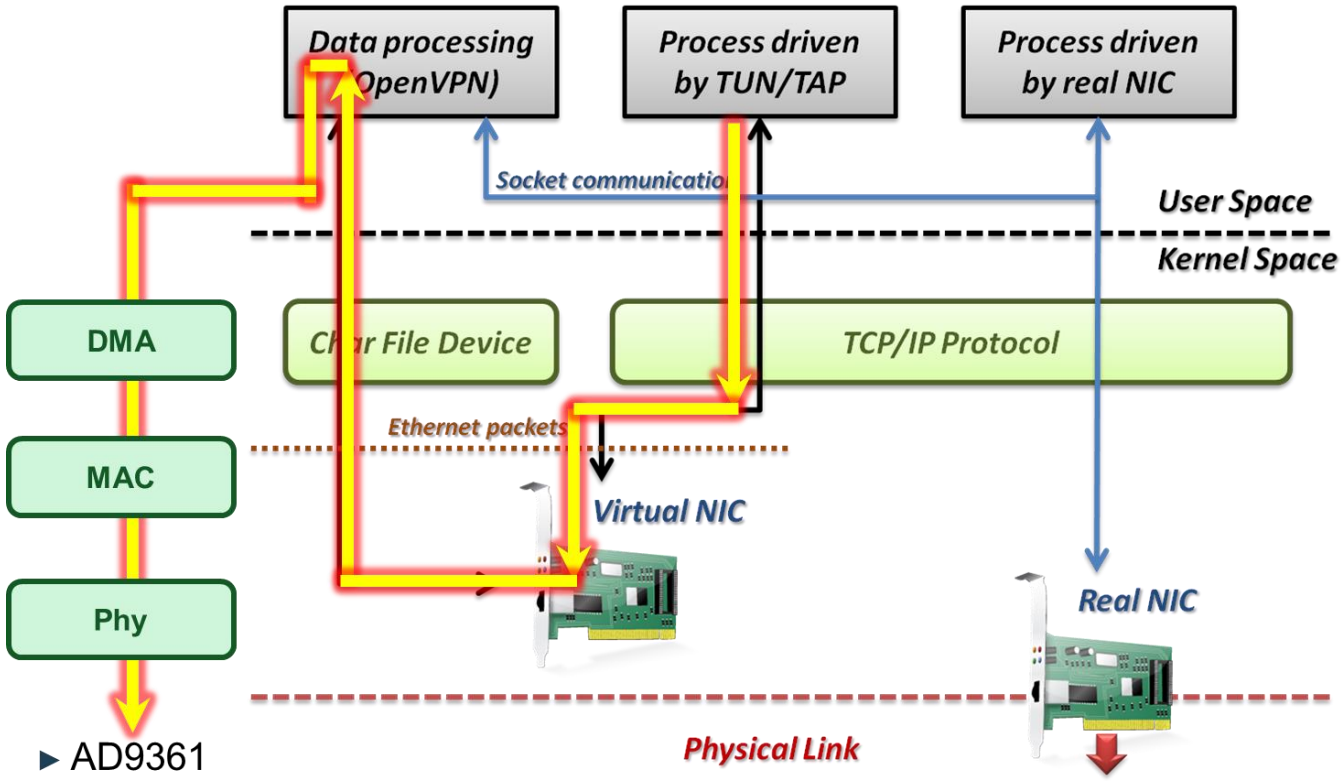
Requirements Specification

ANALOG DEVICES
AHEAD OF WHAT'S POSSIBLE™

MathWorks®

Untethering From MATLAB and Simulink

- **TUN/TAP driver**
  - TUN and TAP are virtual network kernel drivers
    - TAP (as in network tap) simulates an Ethernet device and it operates with layer 2 packets such as Ethernet frames.
    - TUN (as in network TUNnel) simulates a network layer device and it operates with layer 3 packets such as IP.
  - Data flow of TUN/TAP driver
    - Packets sent by an operating system via a TUN/TAP device are delivered to a user-space program that attaches itself to the device.
    - A user-space program may pass packets into a TUN/TAP device. TUN/TAP device delivers (or "injects") these packets to the operating system network stack thus emulating their reception from an external source.



- A user space program can open the tun/tap device just like a file and read and write IP packets from and to it.

ANALOG DEVICES
AHEAD OF WHAT'S POSSIBLE™

MathWorks®

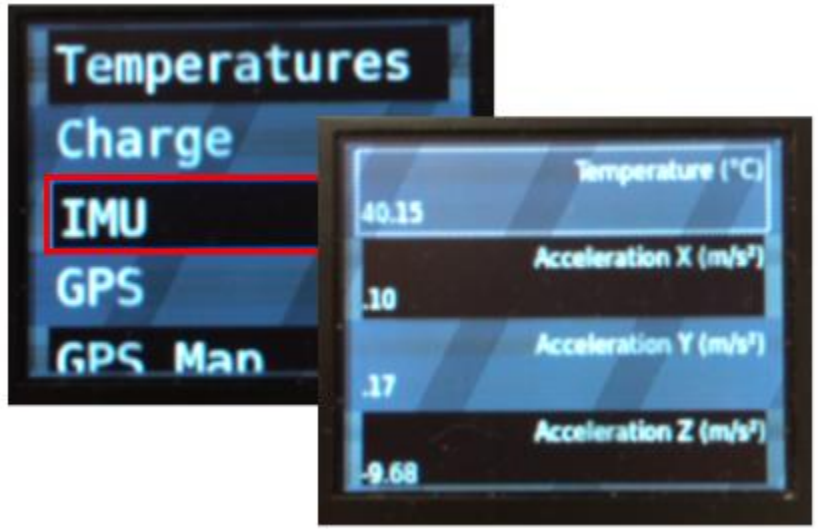
ANALOG DEVICES
AHEAD OF WHAT'S POSSIBLE™

MathWorks®

TUN/TAP In Modem Demo

PackRF GUI application

- Written in Qt
- Control via the PackRF multifunction button
- Radio configuration & status
- Modem configuration & status
- Signal FFT
- GPS and Google maps
- Video link setup & playback
- Video link encryption control
- Access to the system console



ANALOG DEVICES
AHEAD OF WHAT'S POSSIBLE™

MathWorks®

Review

- Integration with custom hardware can prove to be a difficult step
  - New HDL interfaces
  - New HDL design
  - New software interfaces
  - New software infrastructure
  - Different hardware
  - Multiple people with various expertise involved
- Keeping the same flow also for the final design is very beneficial
- The MathWorks custom hardware support flow enables seamless algorithm integration with custom hardware

ANALOG DEVICES
AHEAD OF WHAT'S POSSIBLE™

MathWorks®

Lab 4: Deployed Modem Design

Lab 4 Overview

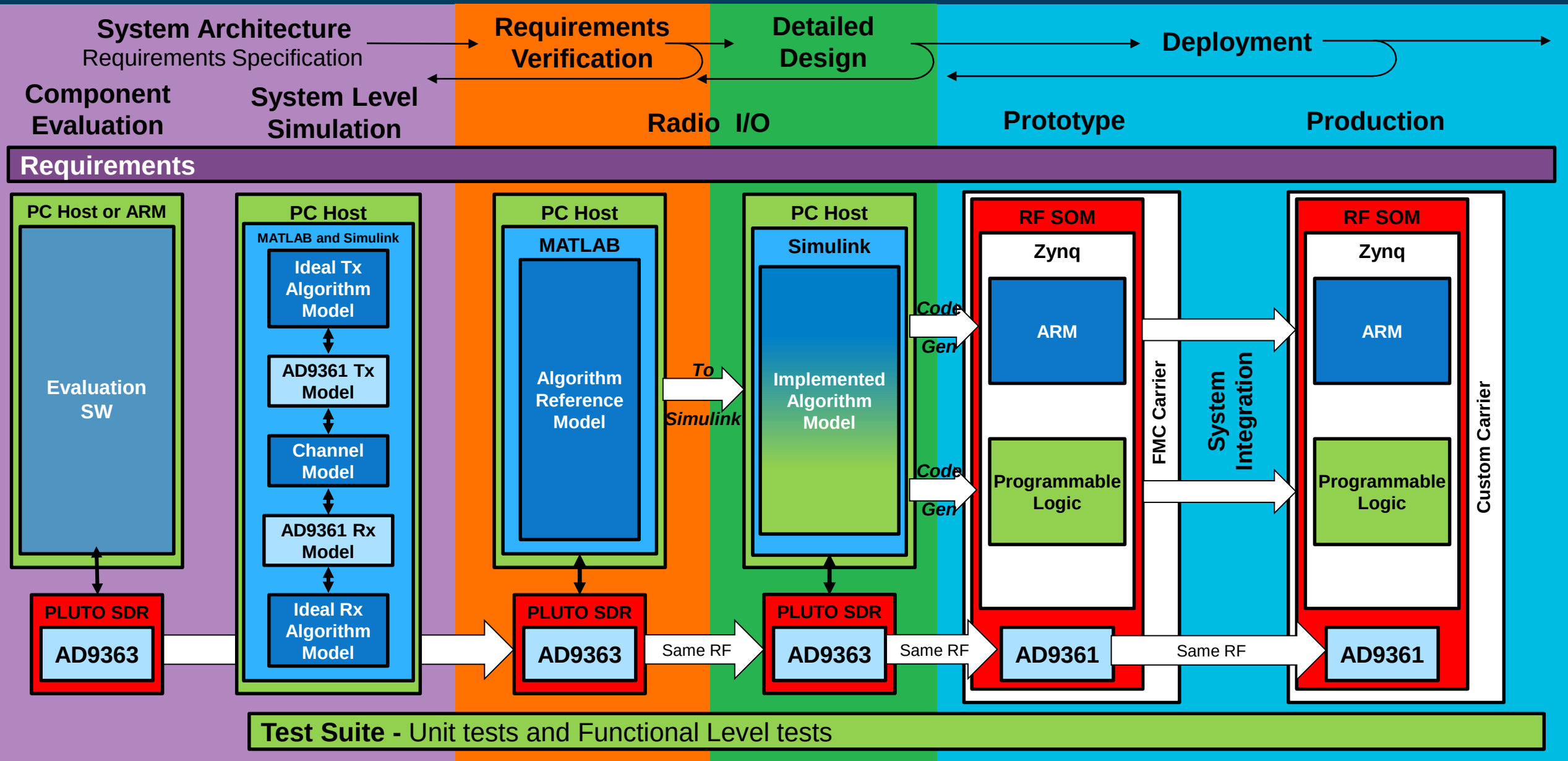
- Lab 4 explores:
  - PackRF device
  - Modem IP deploy to the device
  - Debugging features
  - TUN/TAP interface

ANALOG DEVICES
AHEAD OF WHAT'S POSSIBLE™

MathWorks®

Day 2 review

SDR Workflow



Review

- Single environment for
  - System simulation
  - Algorithm development and validation
  - Requirements verification
  - Data streaming
  - Code generation
- Models shareable across teams / departments
  - Common language to streamline design process
- Target off-the-shelf and custom hardware
- Many features to debug deployed designs from MATLAB and Simulink
- RFSOM is designed to simplify custom hardware designs for deployments
- ADI software and HDL available for RFSOM and similar products help users get to production faster and with less bugs
- ADI software libraries and HDL reference designs are engineered for integration with HDL-Coder generated IP

For more information

- wiki.analog.com
  - Documentation and user guides
  - FPGA reference designs
  - Software drivers
  - ez.analog.com support forum
- mathworks.com/wireless
- mathworks.com/sdr
  - Presentations and videos
  - Request trial software

- When you're to go deeper
  - MathWorks training services
    - Simulink basics
    - Fixed-point Designer
    - HDL Coder
    - Zynq and Zynq SDR targeting

ANALOG DEVICES
AHEAD OF WHAT'S POSSIBLE™

MathWorks®

Thank you