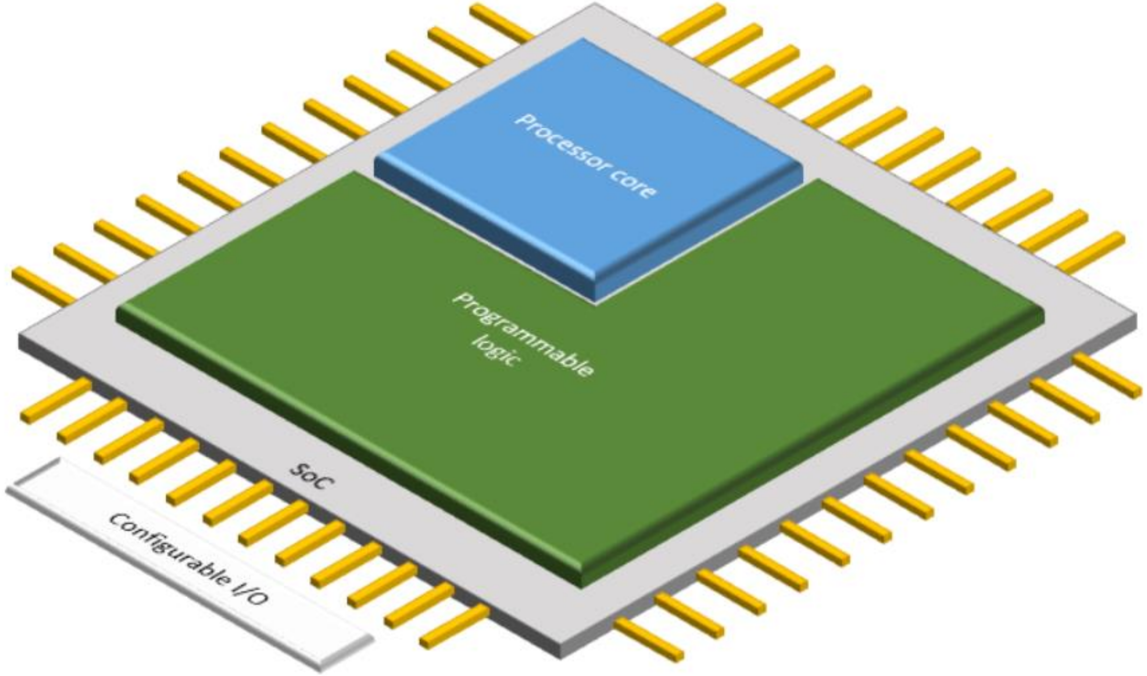
MATLAB EXPO 2018

Designing and Prototyping Digital Systems on SoC FPGA

Hitu Sharma
Application Engineer

Vinod Thomas
Sr. Training Engineer

What is an SoC FPGA?

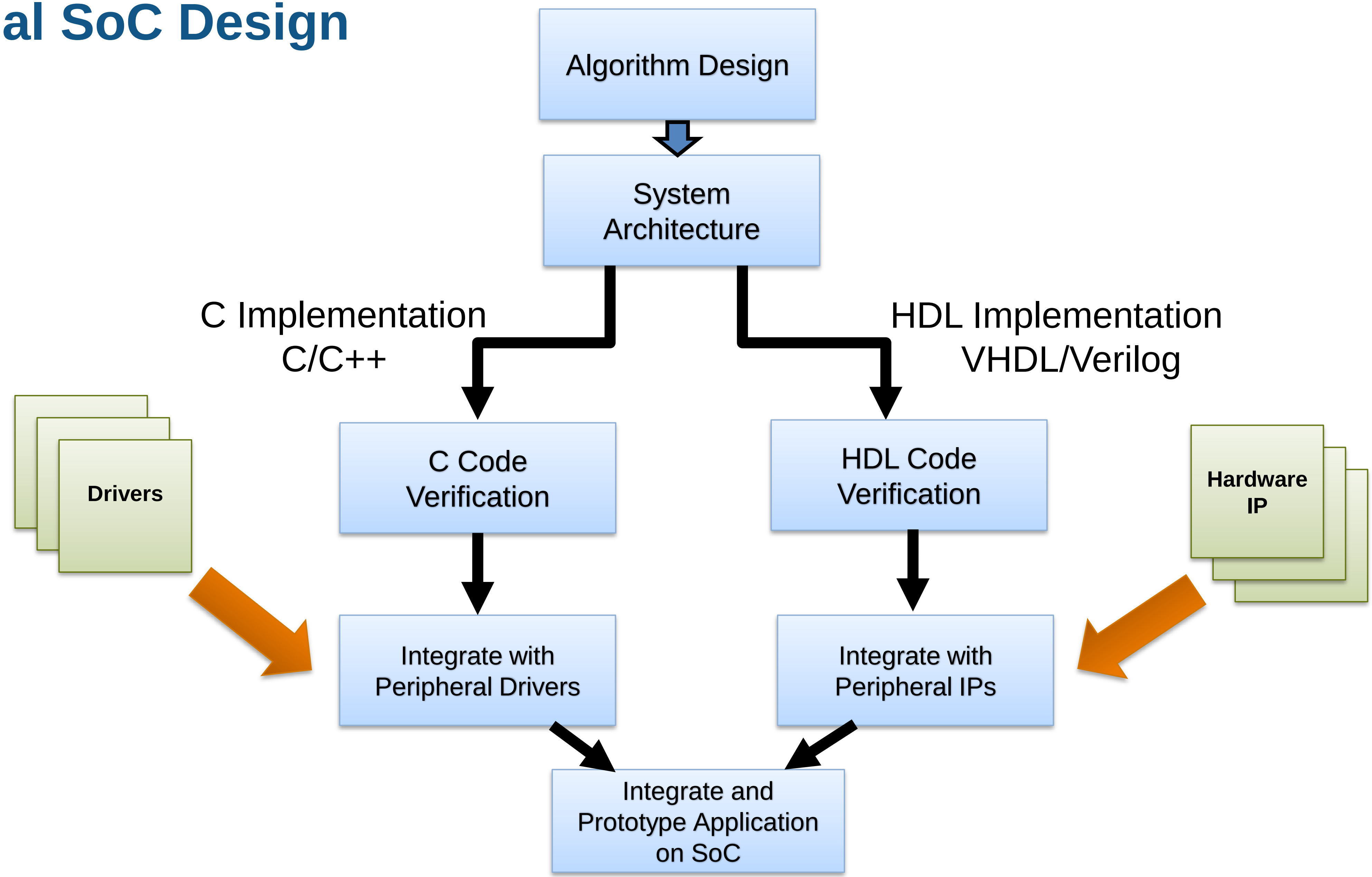


A typical SoC consists of-

- A microcontroller, microprocessor or digital signal processor (DSP) core
- Programmable Logic (FPGA)
- Memory blocks
- External interfaces such as USB, FireWire, Ethernet, USART, SPI, etc.
- Analog interfaces including ADCs and DACs
- Voltage regulators and power management circuits

Combines ***High-speed compute capabilities of FPGAs*** and the ability to perform ***Complex operations on DSPs or MCUs***

Conventional SoC Design Workflow



Algorithm Design

System Architecture

C Implementation
C/C++

HDL Implementation
VHDL/Verilog

Drivers

C Code Verification

HDL Code Verification

Hardware IP

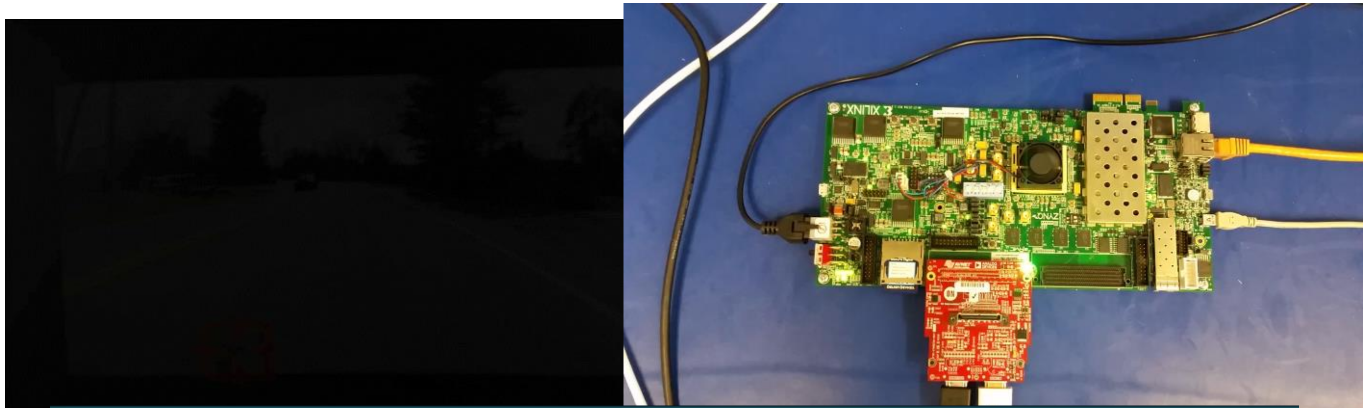
Integrate with Peripheral Drivers

Integrate with Peripheral IPs

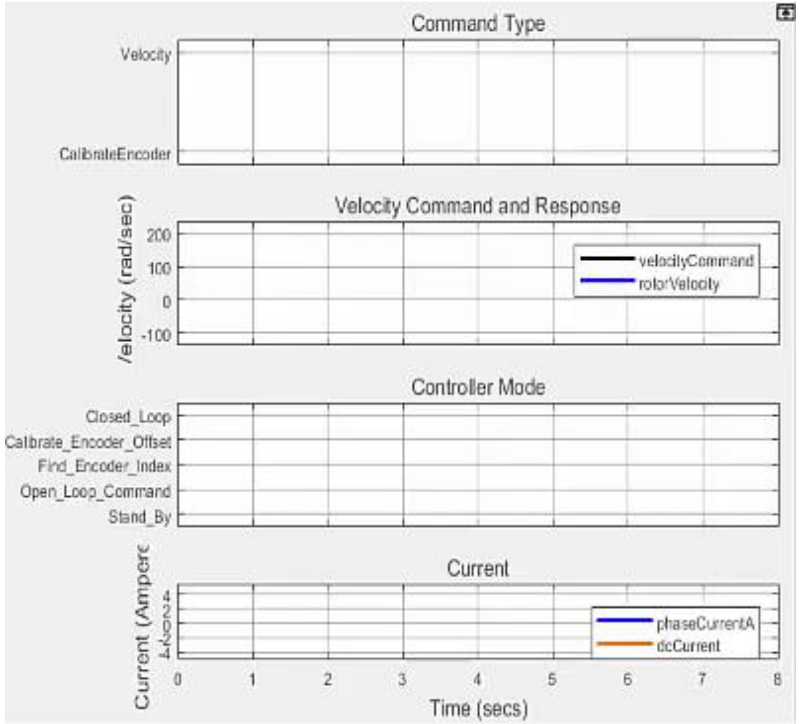
Integrate and Prototype Application on SoC

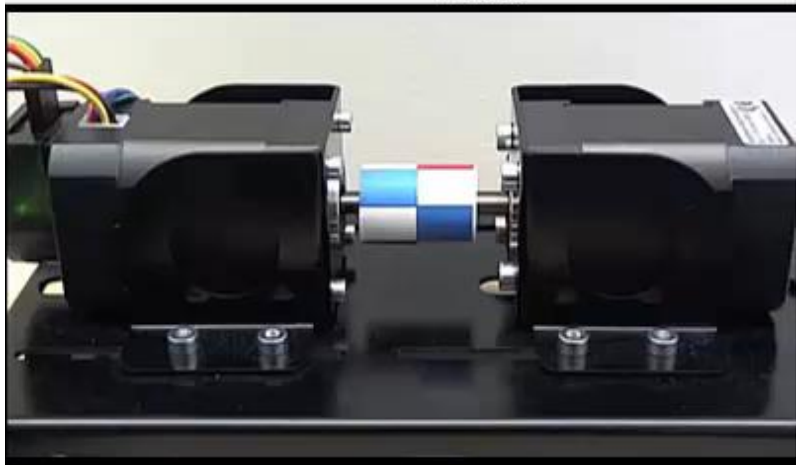
Application Examples

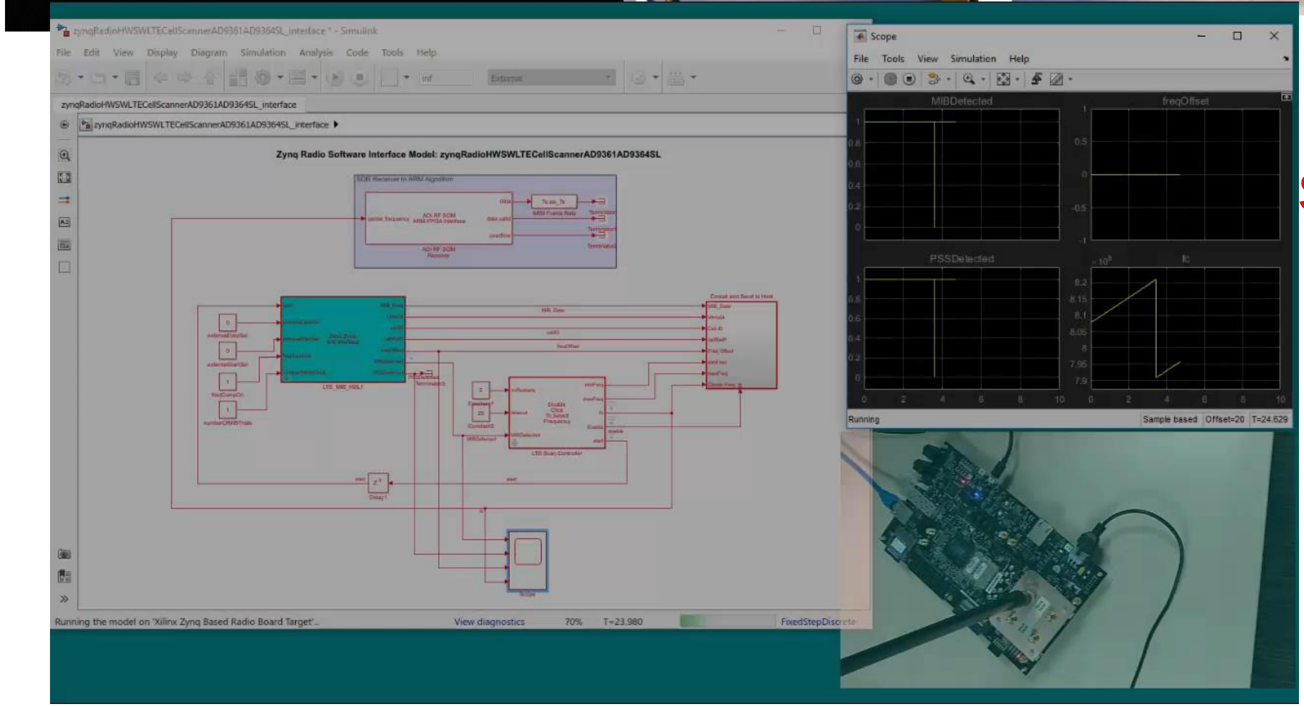
Vision



Motor






zynqRadioHWSWLTECellScannerAD9361AD9364SL_interface * - Simulink
File
Edit
View
Display
Diagram
Simulation
Analysis
Code
Tools
Help
inf
External
zynqRadioHWSWLTECellScannerAD9361AD9364SL_interface
Zynq Radio Software Interface Model: zynqRadioHWSWLTECellScannerAD9361AD9364SL
SDR Receiver to ARM Algorithm
ADI RF SOM ARM-FPGA Interface
Scope
File
Tools
View
Simulation
Help
MIBDetected
freqOffset
PSSDetected
fc
Running
Sample based
Offset=20
T=24.629
Running the model on 'Xilinx Zynq Based Radio Board Target'...
View diagnostics
70%
T=23.980
FixedStepDiscrete

SDR

Customer Case Study: Punch Powertrain

Requirements

- New switched reluctance motor – new complex control strategies
- Fast: 2x the speed of their previous motor
- Target to a Xilinx® Zynq® SoC 7045 device
- Needed to get to market quickly
- No experience designing FPGAs!

✓ Designed integrated E-drive: Motor, power electronics and software

✓ 4 different control strategies implemented

✓ Done in 1.5 years with 2FTE's

✓ Models reusable for production

✓ Smooth integration and validation due to development process – thorough validation before electronics are produced and put in the testbench

"This would not have been possible to design at all previous to adopting HDL code generation from Simulink model-based design"

Challenges in SoC design

Partitioning of Algorithm

- What goes on FPGA and what goes on Processor
- Change in architecture involves reprogramming

Programming and Verification Expertise

- Proficiency in both HDL and C programming
- Both have different verification methodologies
- Late detection of errors

Interface Between Software & Hardware

- Ensure Reliable transfer of data between FPGA and Processor

Verification of the Design

- SoC Verification

Applications & Custom Board

- This involves configuring different peripherals

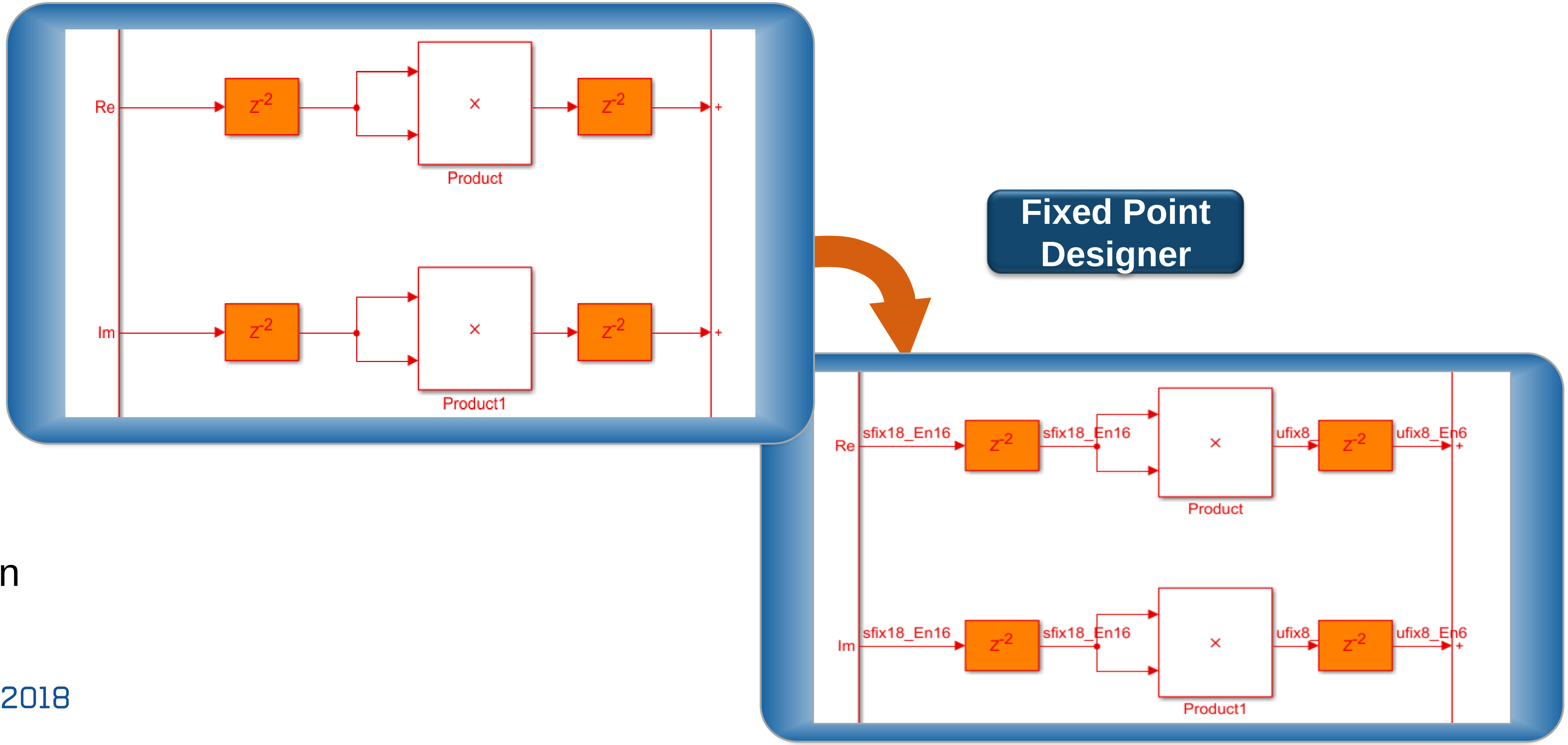
From Behavioral Model to Implementation Model

Simulink

Behavioral algorithm

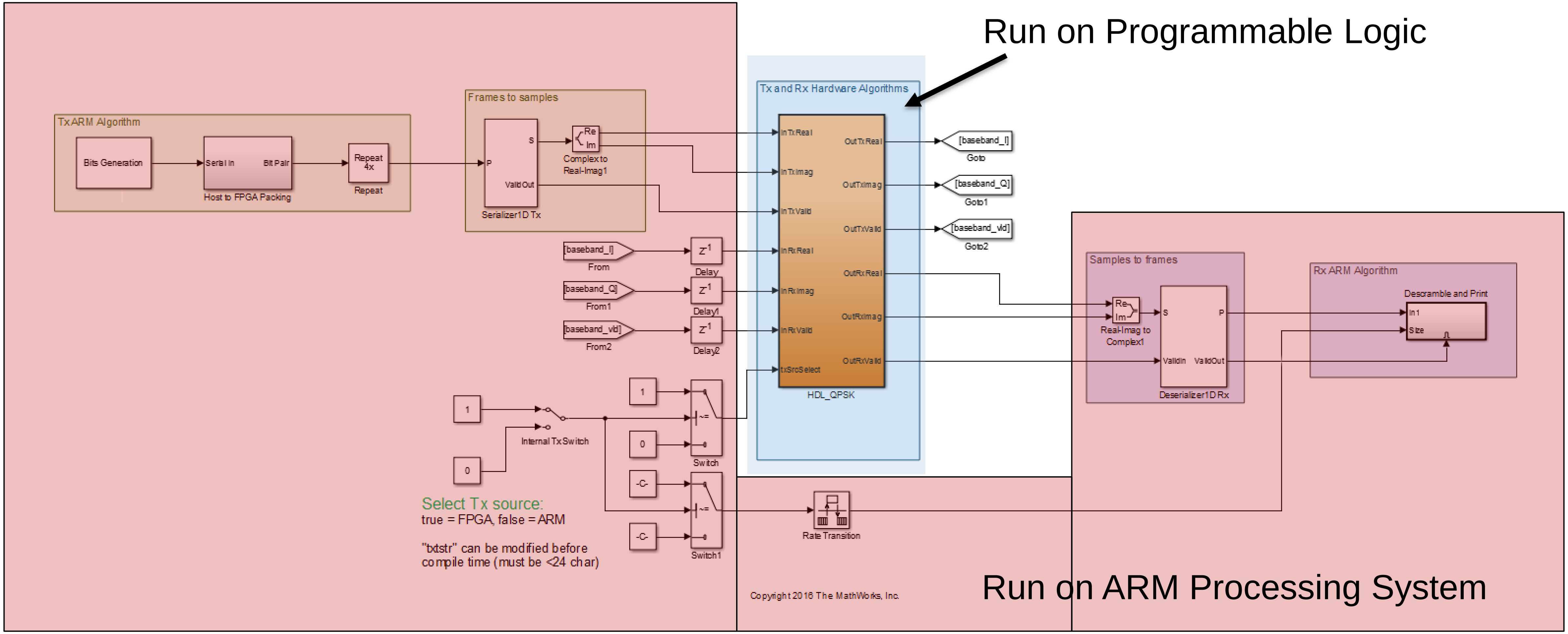
Hardware micro-architecture

Fixed-point implementation



Fixed Point Designer

Target Receiver/Transmitter on Systems-on-Chip (ARM/FPGA)



Select Tx source:
true = FPGA, false = ARM

"txstr" can be modified before compile time (must be <24 char)

Copyright 2016 The MathWorks, Inc.

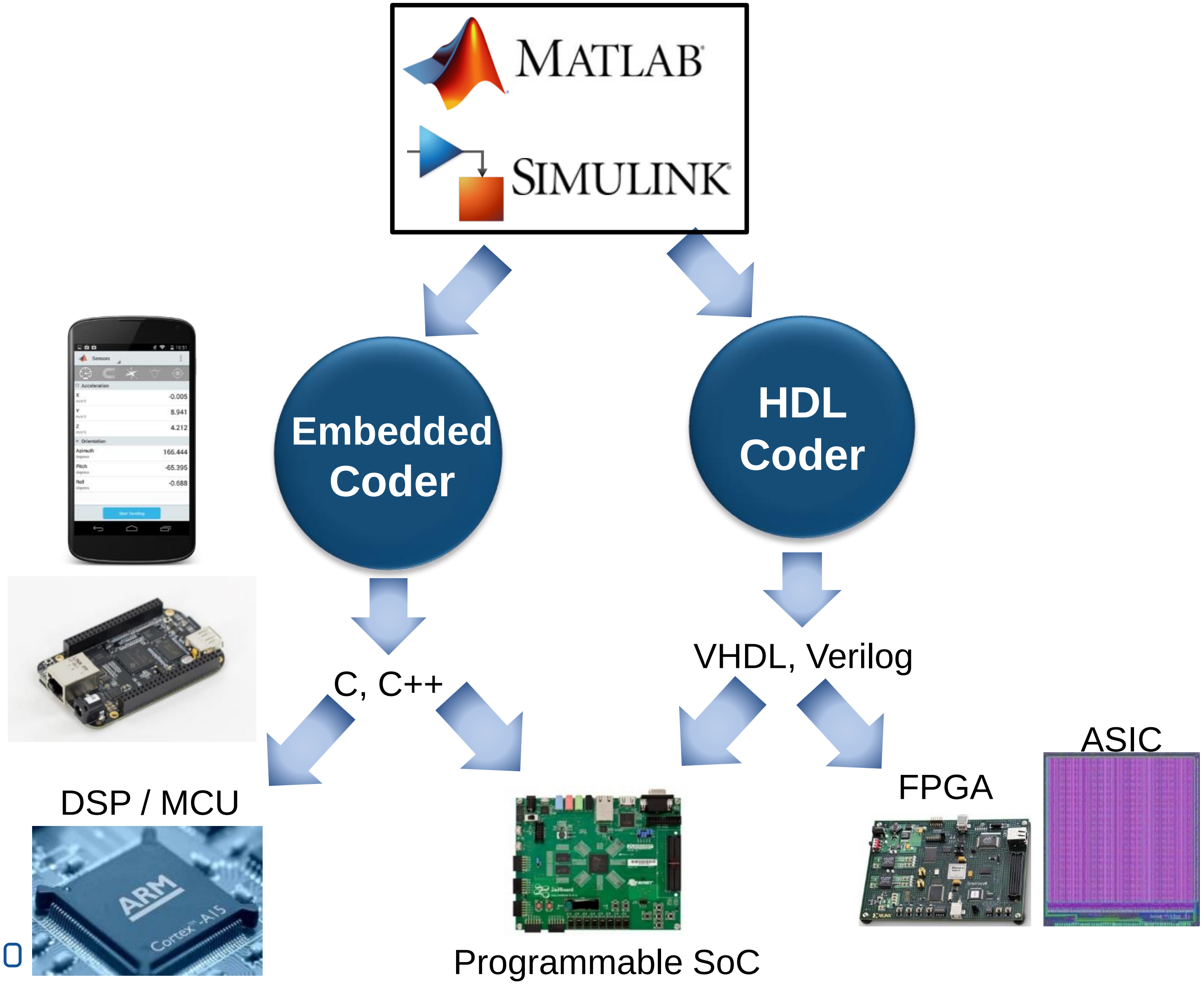
Run on Programmable Logic

Run on ARM Processing System

Addressing Challenges in SoC Design

Partitioning of Algorithm	**Simulink for System Architecture Modeling**
Programming and Verification Expertise	• Proficiency in both HDL and C programming • Both have different verification methodologies • Late detection of errors
Interface Between Software & Hardware	• Ensure Reliable transfer of data between FPGA and Processor
Verification of the Design	• SoC Verification
Applications & Custom Board	• This involves configuring different peripherals

Implement and Prototype Algorithms in Hardware



MATLAB

SIMULINK

Embedded Coder

HDL Coder

C, C++

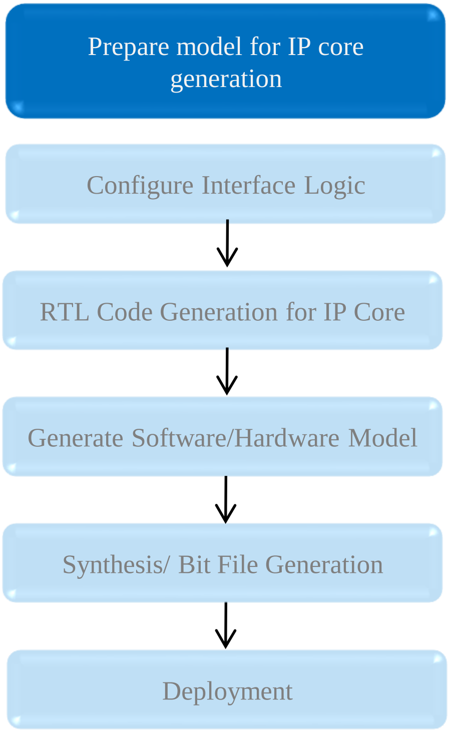
VHDL, Verilog

DSP / MCU

Programmable SoC

FPGA

ASIC



Prepare model for IP core generation

Configure Interface Logic

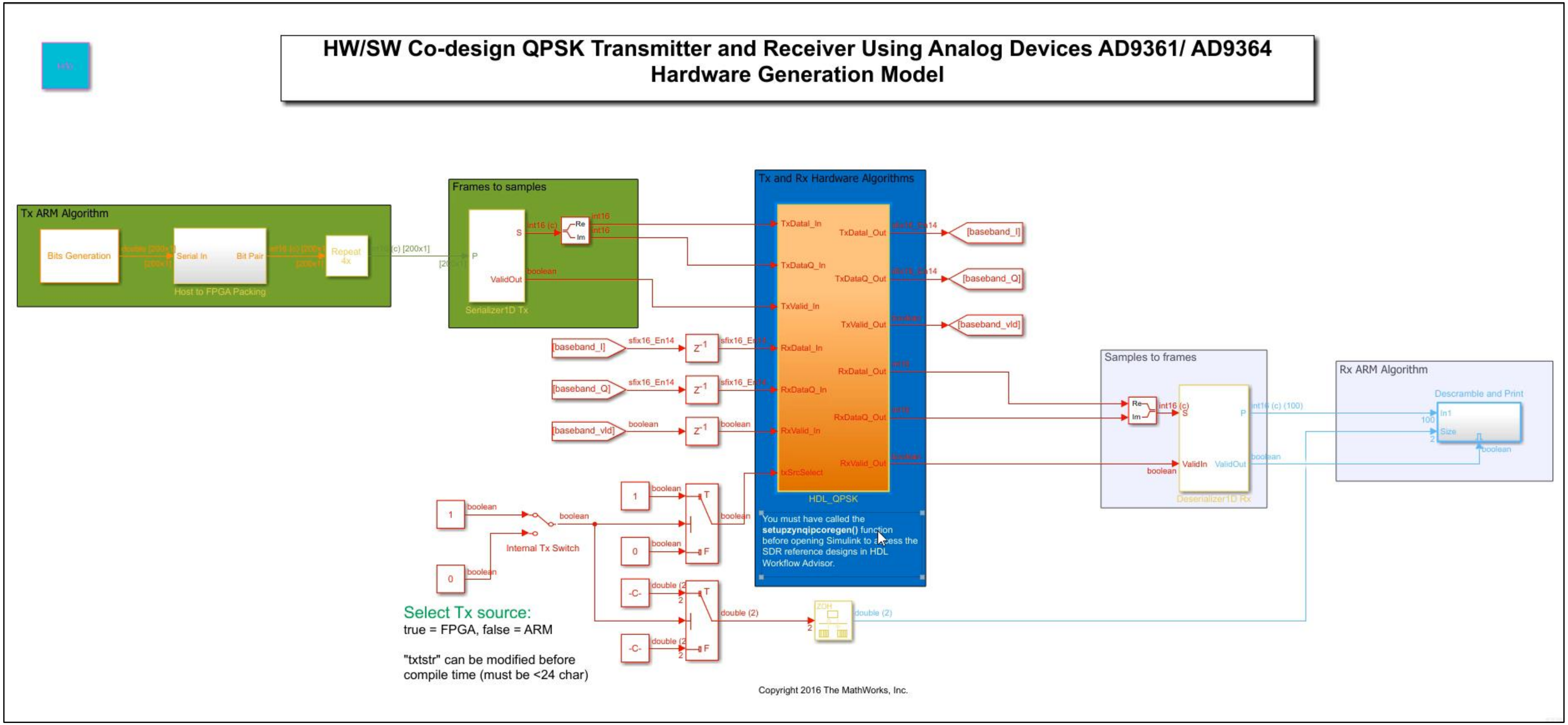
RTL Code Generation for IP Core

Generate Software/Hardware Model

Synthesis/ Bit File Generation

Deployment

HDL code Generation



HW/SW Co-design QPSK Transmitter and Receiver Using Analog Devices AD9361/ AD9364 Hardware Generation Model

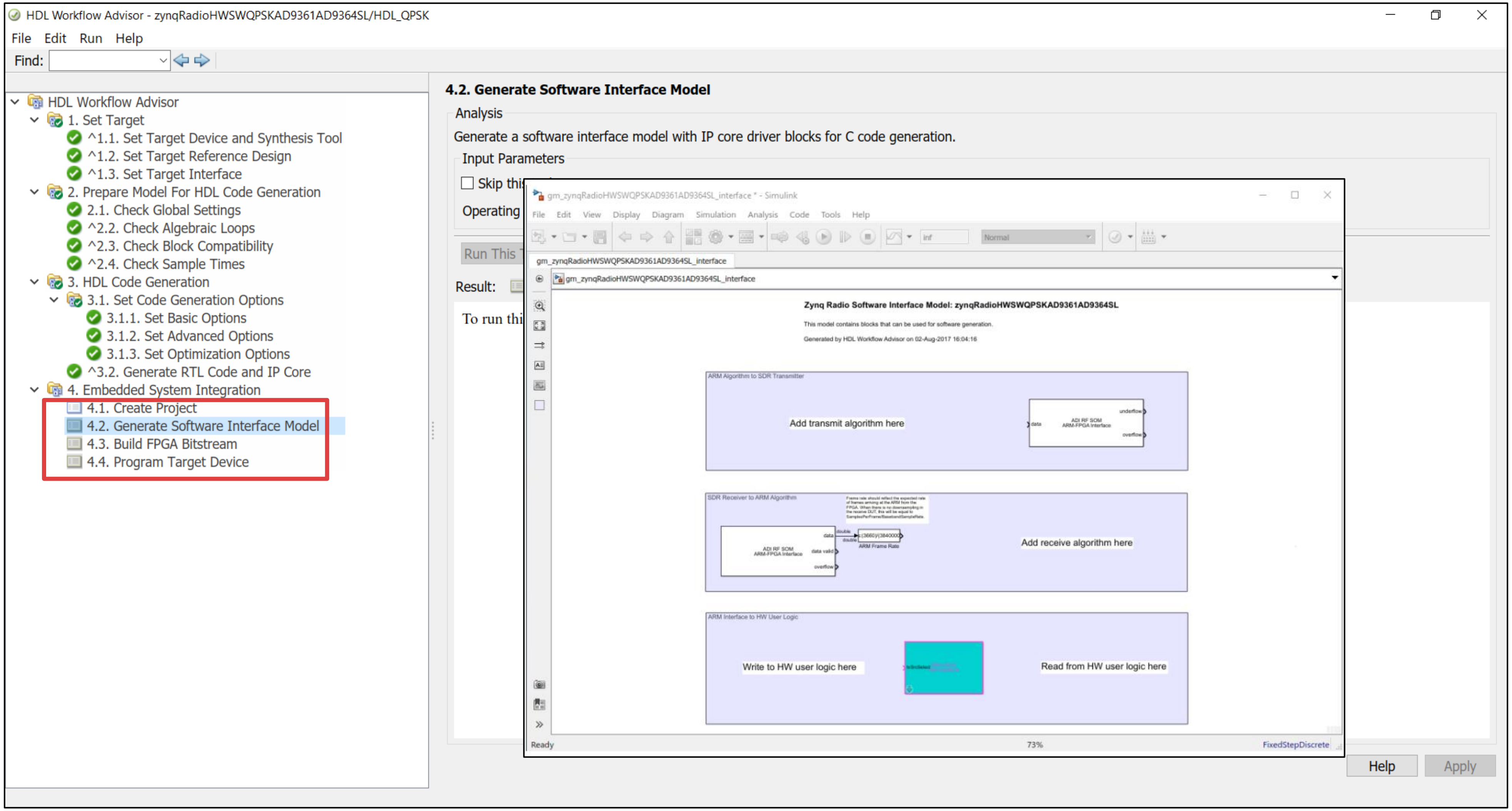
You must have called the **setupzynqipcoregen()** function before opening Simulink to access the SDR reference designs in HDL Workflow Advisor.

Select Tx source:
true = FPGA, false = ARM

"txtstr" can be modified before compile time (must be <24 char)

Copyright 2016 The MathWorks, Inc.

Generate Software Interface model



HDL Workflow Advisor - zynqRadioHWSWQPSKAD9361AD9364SL/HDL_QPSK

- HDL Workflow Advisor
  - 1. Set Target
    - ^1.1. Set Target Device and Synthesis Tool
    - ^1.2. Set Target Reference Design
    - ^1.3. Set Target Interface
  - 2. Prepare Model For HDL Code Generation
    - 2.1. Check Global Settings
    - ^2.2. Check Algebraic Loops
    - ^2.3. Check Block Compatibility
    - ^2.4. Check Sample Times
  - 3. HDL Code Generation
    - 3.1. Set Code Generation Options
      - 3.1.1. Set Basic Options
      - 3.1.2. Set Advanced Options
      - 3.1.3. Set Optimization Options
    - ^3.2. Generate RTL Code and IP Core
  - 4. Embedded System Integration
    - 4.1. Create Project
    - 4.2. Generate Software Interface Model
    - 4.3. Build FPGA Bitstream
    - 4.4. Program Target Device

4.2. Generate Software Interface Model

Analysis

Generate a software interface model with IP core driver blocks for C code generation.

Input Parameters

□ Skip thi

Operating

Run This T

Result:

To run thi

gm_zynqRadioHWSWQPSKAD9361AD9364SL_interface * - Simulink

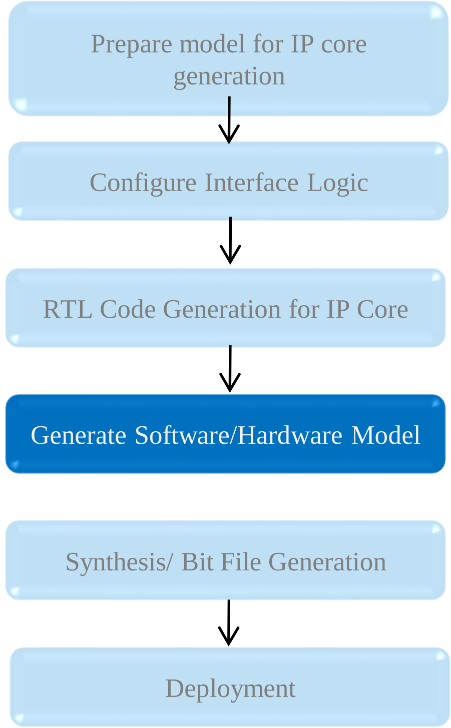
File Edit View Display Diagram Simulation Analysis Code Tools Help
inf
Normal
gm_zynqRadioHWSWQPSKAD9361AD9364SL_interface
gm_zynqRadioHWSWQPSKAD9361AD9364SL_interface

Zynq Radio Software Interface Model: zynqRadioHWSWQPSKAD9361AD9364SL

This model contains blocks that can be used for software generation.

Generated by HDL Workflow Advisor on 02-Aug-2017 16:04:16

ARM Algorithm to SDR Transmitter
Add transmit algorithm here
data
ADI RF SOM
ARM-FPGA Interface
underflow
overflow
SDR Receiver to ARM Algorithm
ADI RF SOM
ARM-FPGA Interface
data
data valid
overflow
double
double
:(3660)/(3840000
ARM Frame Rate
Add receive algorithm here
ARM Interface to HW User Logic
Write to HW user logic here
Read from HW user logic here
Ready
73%
FixedStepDiscrete
Help
Apply



Prepare model for IP core generation

Configure Interface Logic

RTL Code Generation for IP Core

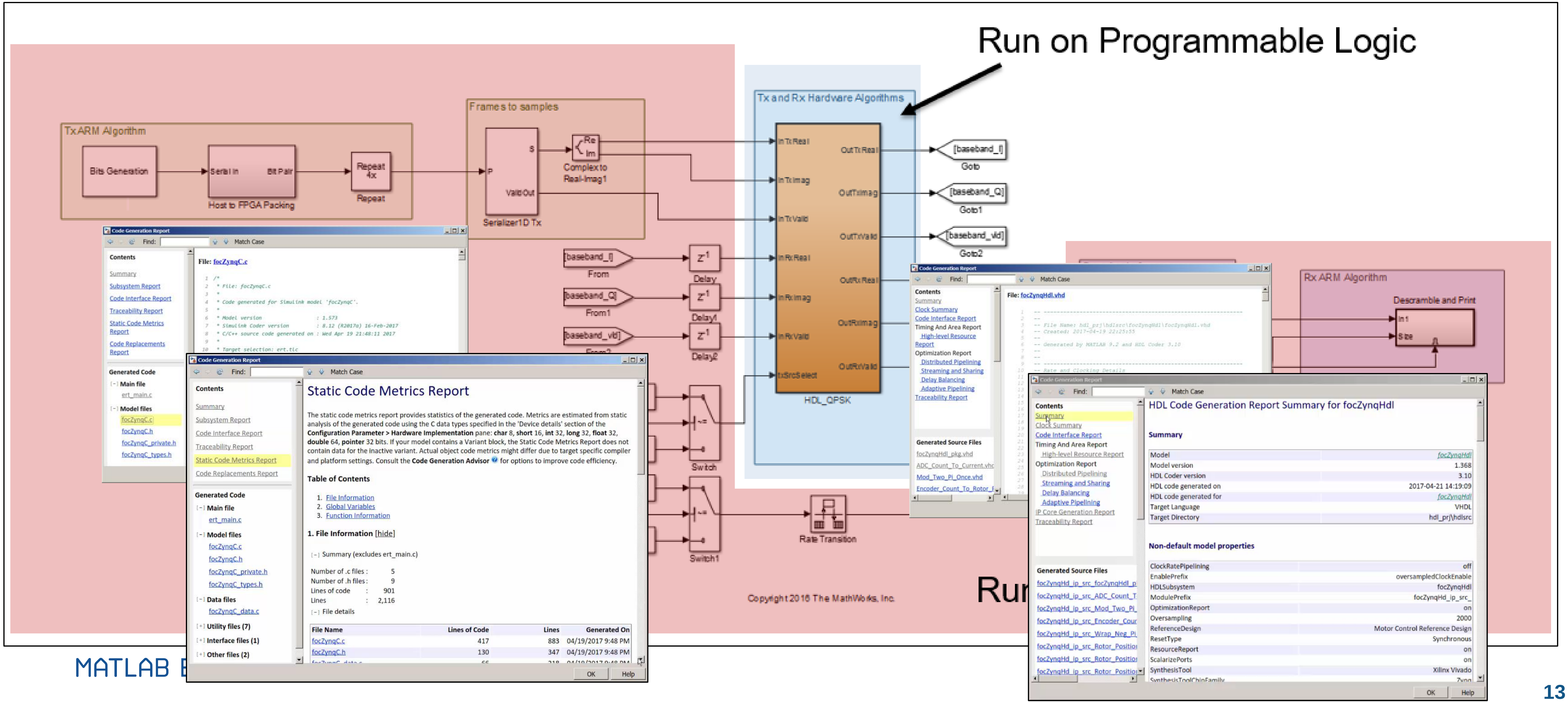
Generate Software/Hardware Model

Synthesis/ Bit File Generation

Deployment

MATLAB EXPO 2018

Algorithm Partitioning and HW-SW Co-Design



Run on Programmable Logic

Run

Code Generation Report

Find: Match Case

Contents

- Summary
- Subsystem Report
- Code Interface Report
- Traceability Report
- Static Code Metrics Report
- Code Replacements Report

Generated Code

[-] Main file

- ert_main.c

[-] Model files

- focZynqC.c
- focZynqC.h
- focZynqC_private.h
- focZynqC_types.h

File: focZynqC.c

```
/*
 * File: focZynqC.c
 *
 * Code generated for Simulink model 'focZynqC'.
 *
 * Model version                  : 1.573
 * Simulink Coder version         : 8.12 (R2017a) 16-Feb-2017
 * C/C++ source code generated on : Wed Apr 19 21:48:11 2017
 *
 * Target selection: ert.tlc
```

Code Generation Report

Find: Match Case

Contents

- Summary
- Subsystem Report
- Code Interface Report
- Traceability Report
- Static Code Metrics Report
- Code Replacements Report

Generated Code

[-] Main file

- ert_main.c

[-] Model files

- focZynqC.c
- focZynqC.h
- focZynqC_private.h
- focZynqC_types.h

[-] Data files

- focZynqC_data.c

[+] Utility files (7)

[+] Interface files (1)

[+] Other files (2)

Static Code Metrics Report

The static code metrics report provides statistics of the generated code. Metrics are estimated from static analysis of the generated code using the C data types specified in the 'Device details' section of the **Configuration Parameter > Hardware Implementation** pane: **char** 8, **short** 16, **int** 32, **long** 32, **float** 32, **double** 64, **pointer** 32 bits. If your model contains a Variant block, the Static Code Metrics Report does not contain data for the inactive variant. Actual object code metrics might differ due to target specific compiler and platform settings. Consult the **Code Generation Advisor** for options to improve code efficiency.

Table of Contents

1. File Information
2. Global Variables
3. Function Information

1. File Information [hide]

[-] Summary (excludes ert_main.c)

```
Number of .c files :     5
Number of .h files :     9
Lines of code      :   901
Lines              : 2,116
```

[-] File details

File Name	Lines of Code	Lines	Generated On
focZynqC.c	417	883	04/19/2017 9:48 PM
focZynqC.h	130	347	04/19/2017 9:48 PM
focZynqC_data.c	66	218	04/19/2017 9:48 PM

OK Help

Code Generation Report

Find: Match Case

Contents

- Summary
- Clock Summary
- Code Interface Report
- Timing And Area Report
  - High-level Resource Report
- Optimization Report
  - Distributed Pipelining
  - Streaming and Sharing
  - Delay Balancing
  - Adaptive Pipelining
- Traceability Report

Generated Source Files

- focZynqHdl_pkg.vhd
- ADC_Count_To_Current.vhd
- Mod_Two_Pi_Once.vhd
- Encoder_Count_To_Rotor_P

File: focZynqHdl.vhd

```
-- -------------------------------------------------------------
--
-- File Name: hdl_prj\hdlsrc\focZynqHdl\focZynqHdl.vhd
-- Created: 2017-04-19 22:25:55
--
-- Generated by MATLAB 9.2 and HDL Coder 3.10
--
--
-- -------------------------------------------------------------
-- Rate and Clocking Details
```

Code Generation Report

Find: Match Case

Contents

- Summary
- Clock Summary
- Code Interface Report
- Timing And Area Report
  - High-level Resource Report
- Optimization Report
  - Distributed Pipelining
  - Streaming and Sharing
  - Delay Balancing
  - Adaptive Pipelining
- IP Core Generation Report
- Traceability Report

Generated Source Files

- focZynqHd_ip_src_focZynqHdl_p
- focZynqHd_ip_src_ADC_Count_T
- focZynqHd_ip_src_Mod_Two_Pi_
- focZynqHd_ip_src_Encoder_Cour
- focZynqHd_ip_src_Wrap_Neg_Pi_
- focZynqHd_ip_src_Rotor_Positio
- focZynqHd_ip_src_Rotor_Positio
- focZynqHd_ip_src_Rotor_Positio

HDL Code Generation Report Summary for focZynqHdl

Summary

Model	focZynqHdl
Model version	1.368
HDL Coder version	3.10
HDL code generated on	2017-04-21 14:19:09
HDL code generated for	focZynqHdl
Target Language	VHDL
Target Directory	hdl_prj\hdlsrc

Non-default model properties

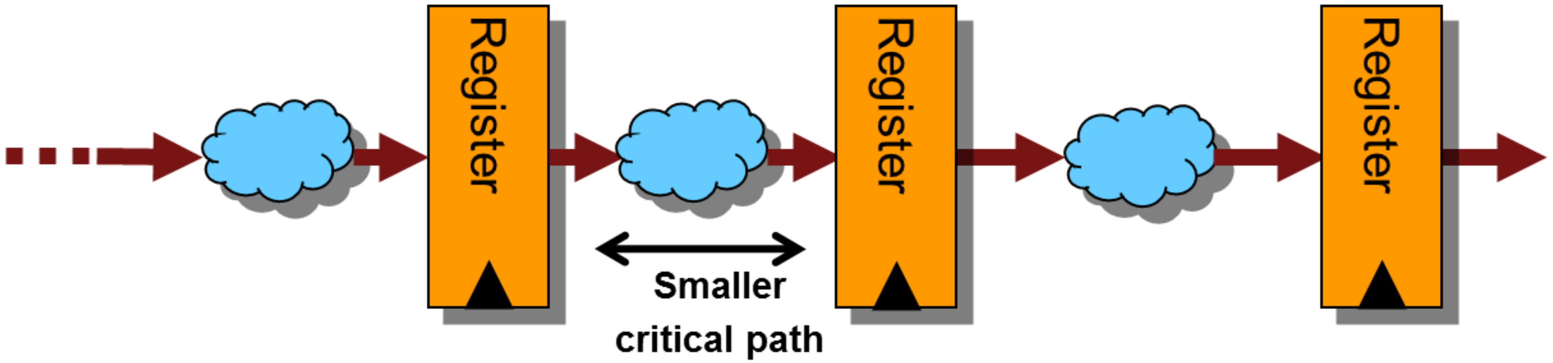
ClockRatePipelining	off
EnablePrefix	oversampledClockEnable
HDLSubsystem	focZynqHdl
ModulePrefix	focZynqHd_ip_src_
OptimizationReport	on
Oversampling	2000
ReferenceDesign	Motor Control Reference Design
ResetType	Synchronous
ResourceReport	on
ScalarizePorts	on
SynthesisTool	Xilinx Vivado
SynthesisToolChipFamily	Zynq

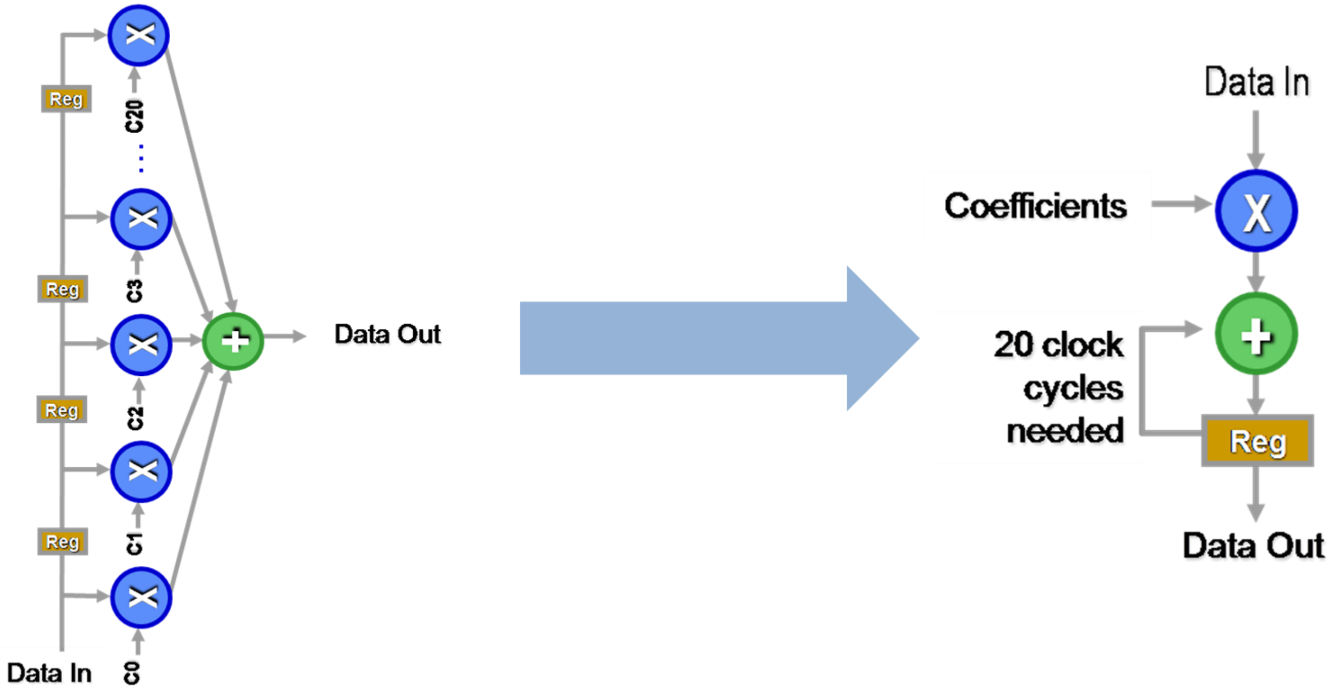
OK Help

HDL Optimizations

- **Speed Optimization**
  - Distributed Pipelining
  - Adaptive Pipelining
  - Clock-Rate Pipelining

- **Area Optimization**
  - Streaming
  - RAM Mapping
  - Resource Sharing



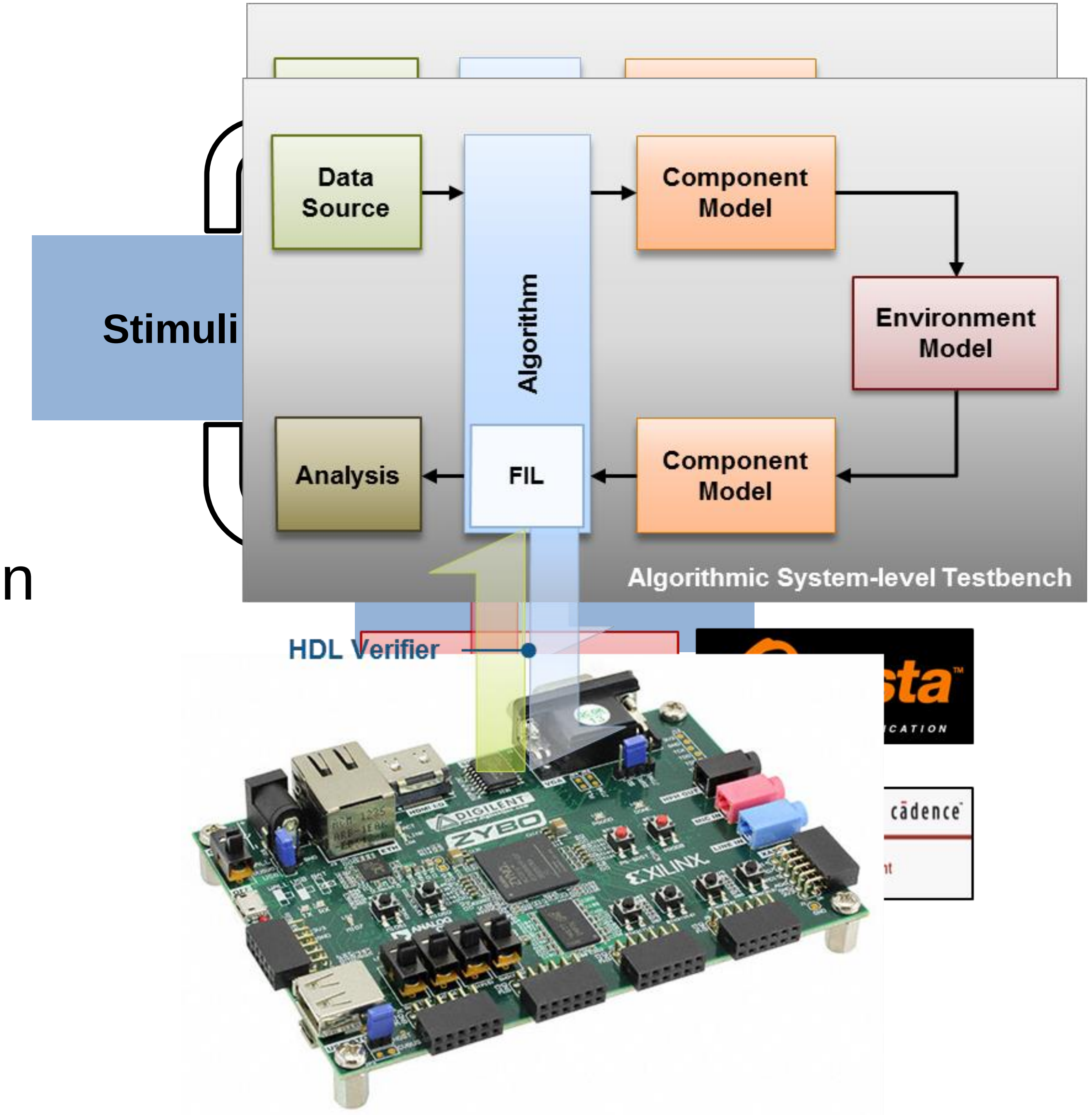


Hardware Verification

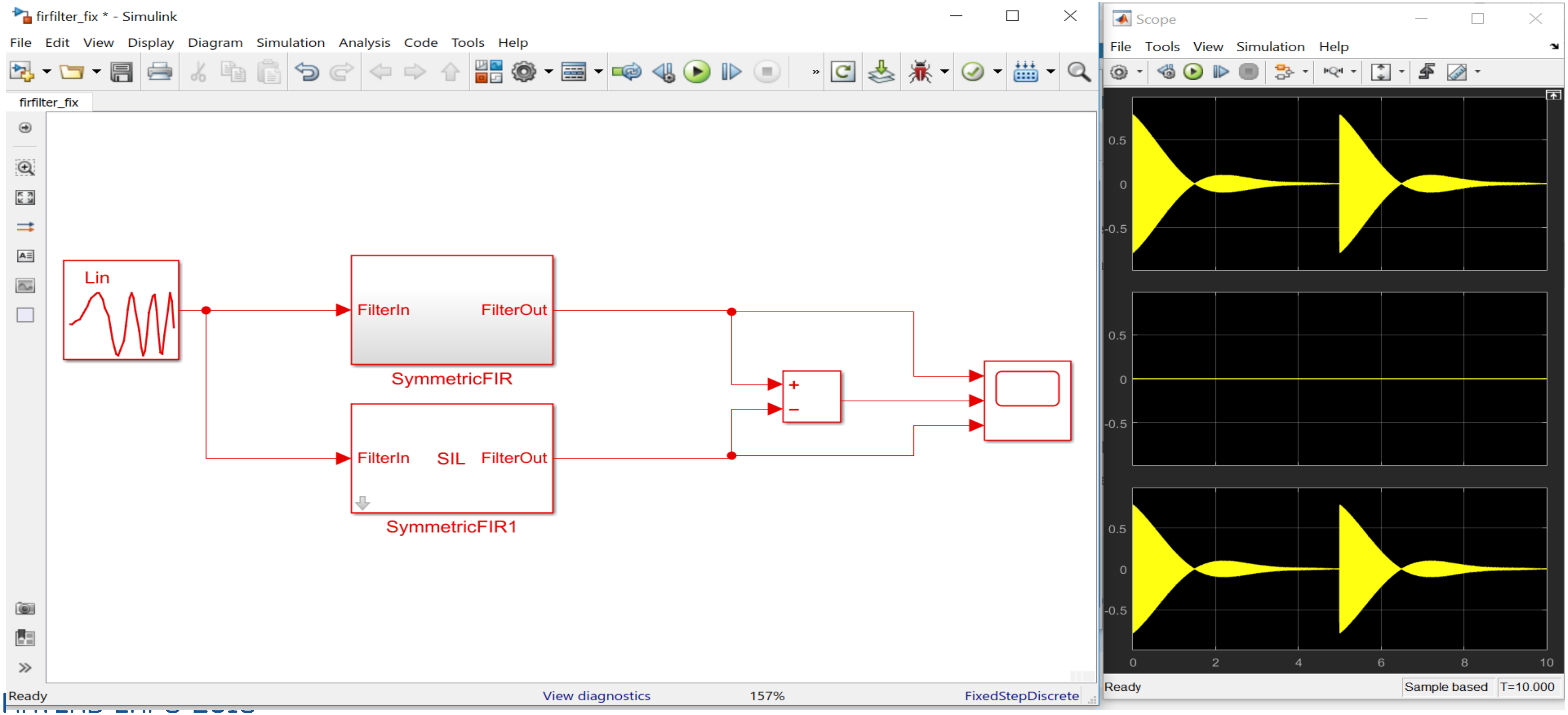
  - “Validation Model” generation by HDL Coder

- RTL code against Implementation Model
  - HDL Cosimulation through HDL Verifier

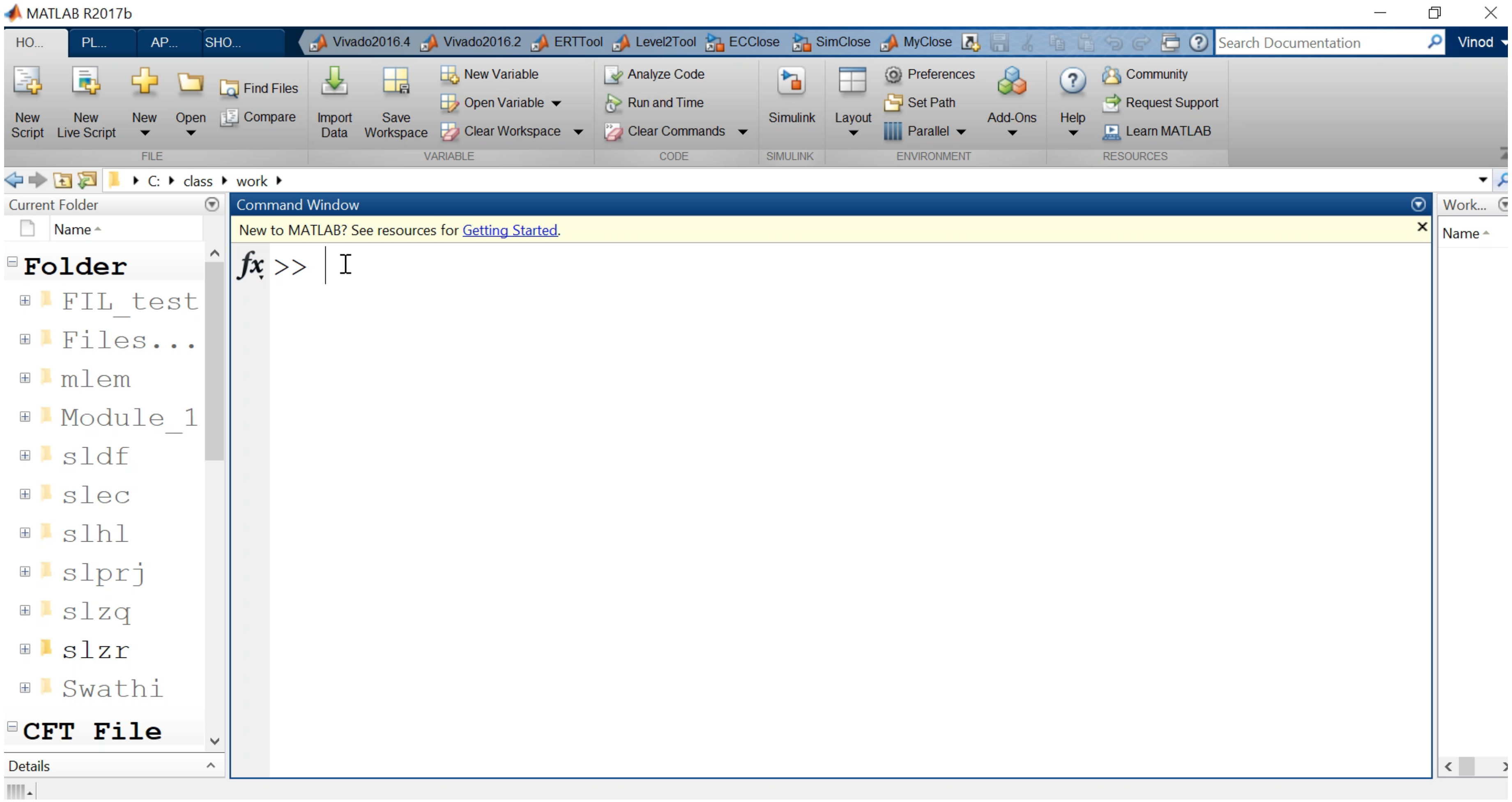
- Hardware Results against Implementation Model
  - FPGA-in-the-loop verification through HDL Verifier



Software in the Loop (SIL) Verification



HDL Library Summary



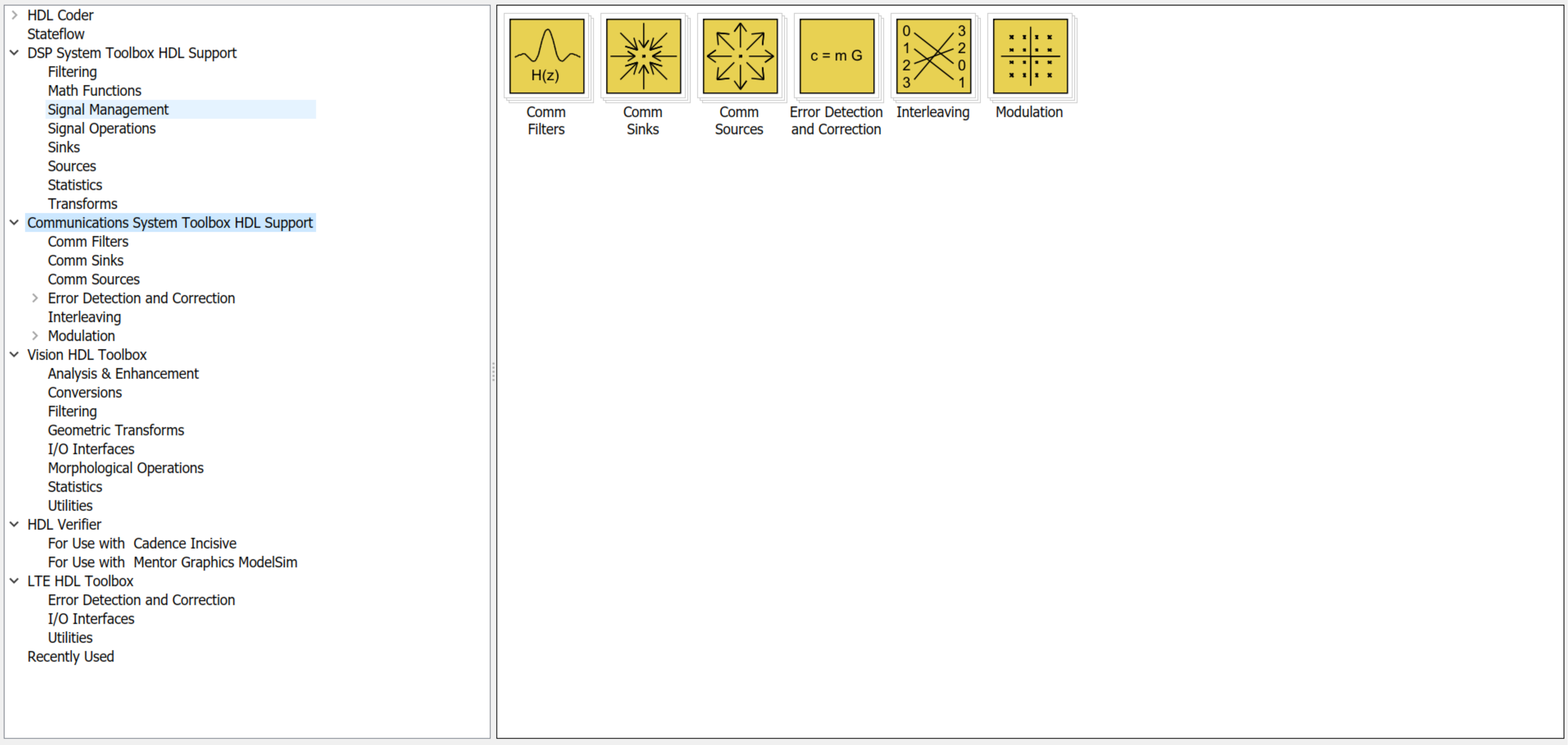
New to MATLAB? See resources for Getting Started.

```
fx >>
```

Folder
- FIL_test
- Files...
- mlem
- Module_1
- sldf
- slec
- slhl
- slprj
- slzq
- slzr
- Swathi

CFT File

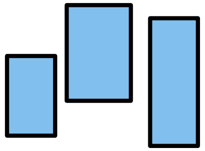
HDL Library Summary



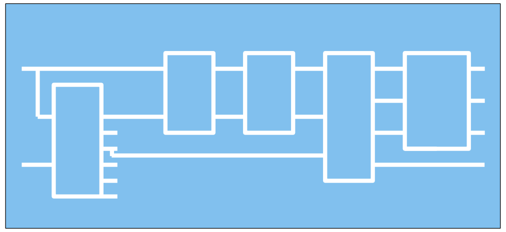
- HDL Coder
- Stateflow
- DSP System Toolbox HDL Support
  - Filtering
  - Math Functions
  - Signal Management
  - Signal Operations
  - Sinks
  - Sources
  - Statistics
  - Transforms
- Communications System Toolbox HDL Support
  - Comm Filters
  - Comm Sinks
  - Comm Sources
  - Error Detection and Correction
  - Interleaving
  - Modulation
- Vision HDL Toolbox
  - Analysis & Enhancement
  - Conversions
  - Filtering
  - Geometric Transforms
  - I/O Interfaces
  - Morphological Operations
  - Statistics
  - Utilities
- HDL Verifier
  - For Use with Cadence Incisive
  - For Use with Mentor Graphics ModelSim
- LTE HDL Toolbox
  - Error Detection and Correction
  - I/O Interfaces
  - Utilities
- Recently Used

LTE HDL Toolbox R2017b

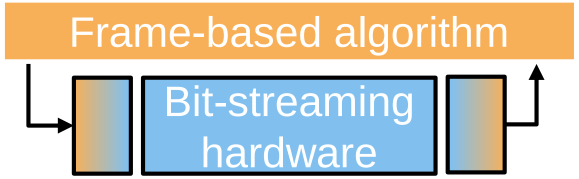
LTE HDL-optimized blocks



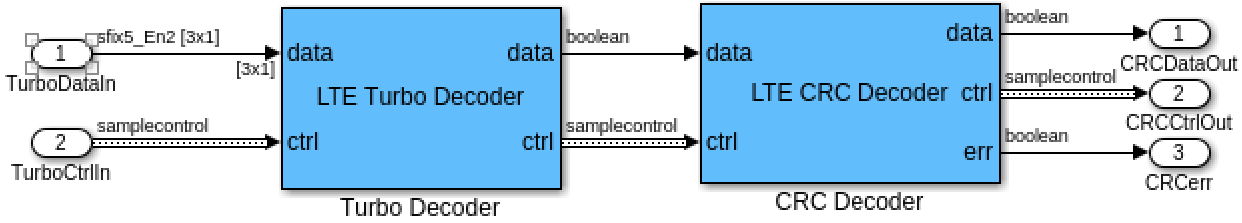
Reference applications



Utilities



- Turbo Encoder
- Turbo Decoder
- Convolutional Encoder
- Convolutional Decoder
- CRC Encoder
- CRC Decoder



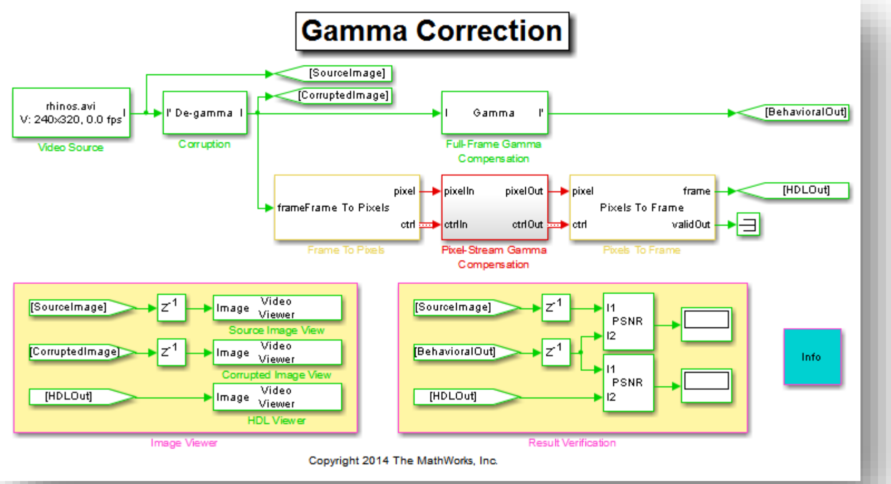
- PSS/SSS Detection
- MIB Recovery
- LTE Frequency Scanner

- Frame-to-samples / samples-to-frame
- Sample bus creator / selector
- Templates to connect MATLAB tests/golden reference to Simulink HW implementation

Vision HDL Toolbox

Design and prototype video image processing systems

- Modeling hardware behavior of the algorithms
  - Pixel-based functions and blocks
  - Conversion between frames and pixels
  - Standard and custom frame sizes

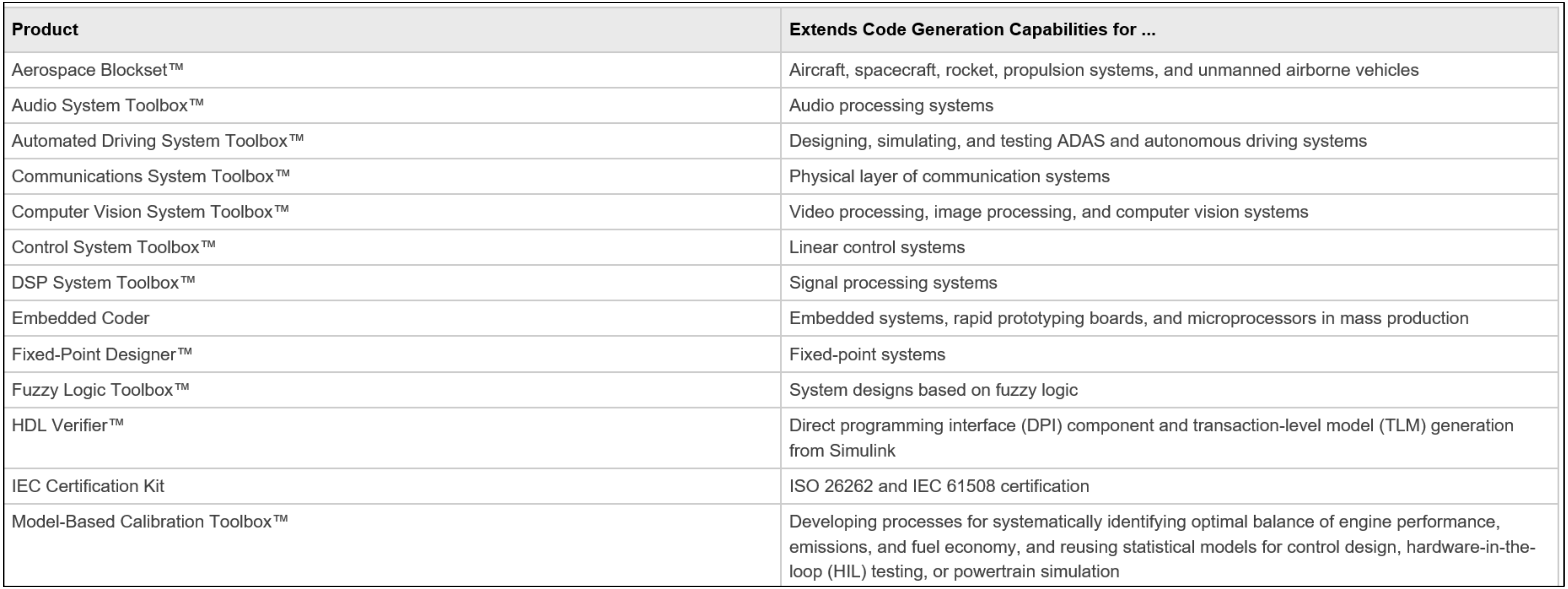


- Prototyping algorithms on hardware

  (**With HDL Coder**) Efficient and readable HDL code

  (**With HDL Verifier**) FPGA-in-the-loop testing and acceleration

C code supported libraries

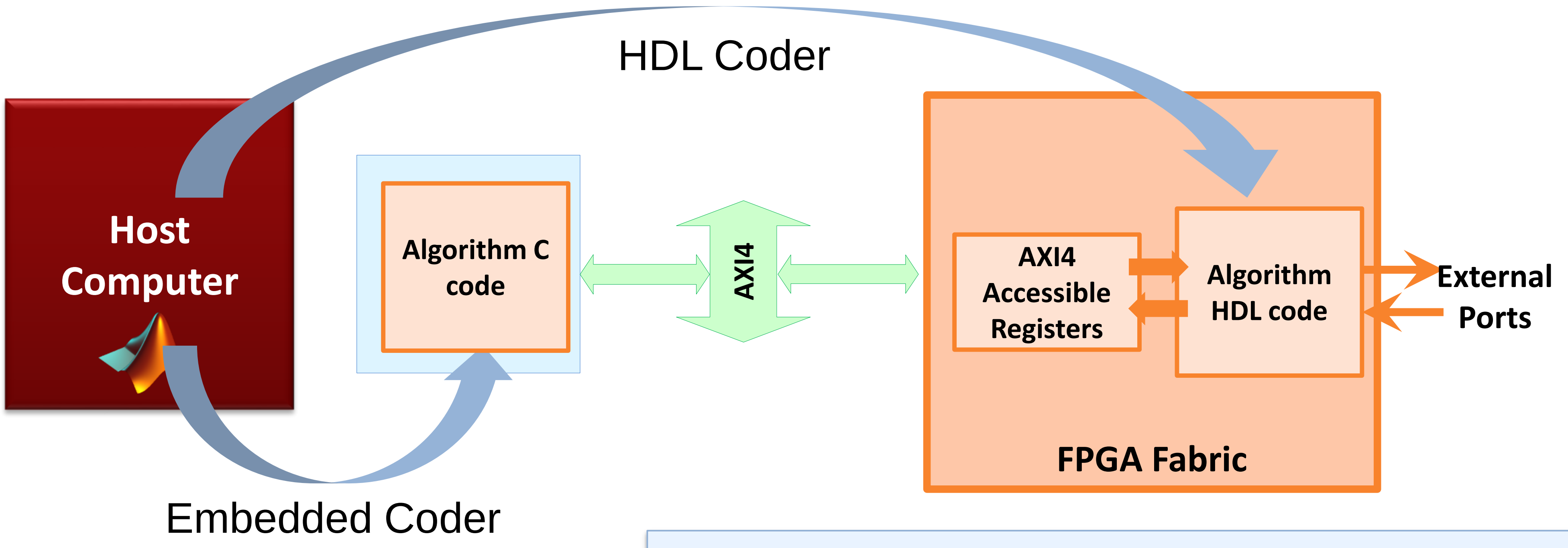


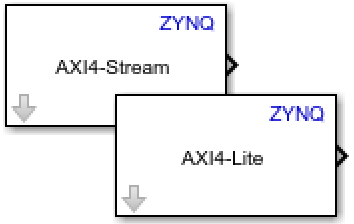
Product	Extends Code Generation Capabilities for ...
Aerospace Blockset™	Aircraft, spacecraft, rocket, propulsion systems, and unmanned airborne vehicles
Audio System Toolbox™	Audio processing systems
Automated Driving System Toolbox™	Designing, simulating, and testing ADAS and autonomous driving systems
Communications System Toolbox™	Physical layer of communication systems
Computer Vision System Toolbox™	Video processing, image processing, and computer vision systems
Control System Toolbox™	Linear control systems
DSP System Toolbox™	Signal processing systems
Embedded Coder	Embedded systems, rapid prototyping boards, and microprocessors in mass production
Fixed-Point Designer™	Fixed-point systems
Fuzzy Logic Toolbox™	System designs based on fuzzy logic
HDL Verifier™	Direct programming interface (DPI) component and transaction-level model (TLM) generation from Simulink
IEC Certification Kit	ISO 26262 and IEC 61508 certification
Model-Based Calibration Toolbox™	Developing processes for systematically identifying optimal balance of engine performance, emissions, and fuel economy, and reusing statistical models for control design, hardware-in-the-loop (HIL) testing, or powertrain simulation

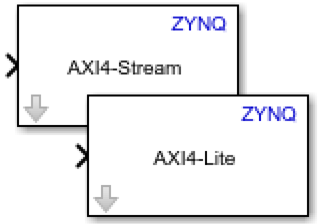
Addressing Challenges in SoC Design

Partitioning of Algorithm	**Simulink for System Architecture Modeling**
Programming and Verification Expertise	**HDL and Embedded Coder**
Interface Between Software & Hardware	• Ensure Reliable transfer of data between FPGA and Processor
Verification of the Design	• SoC Verification
Applications & Custom Board	• This involves configuring different peripherals

Interface Between Hardware and Software



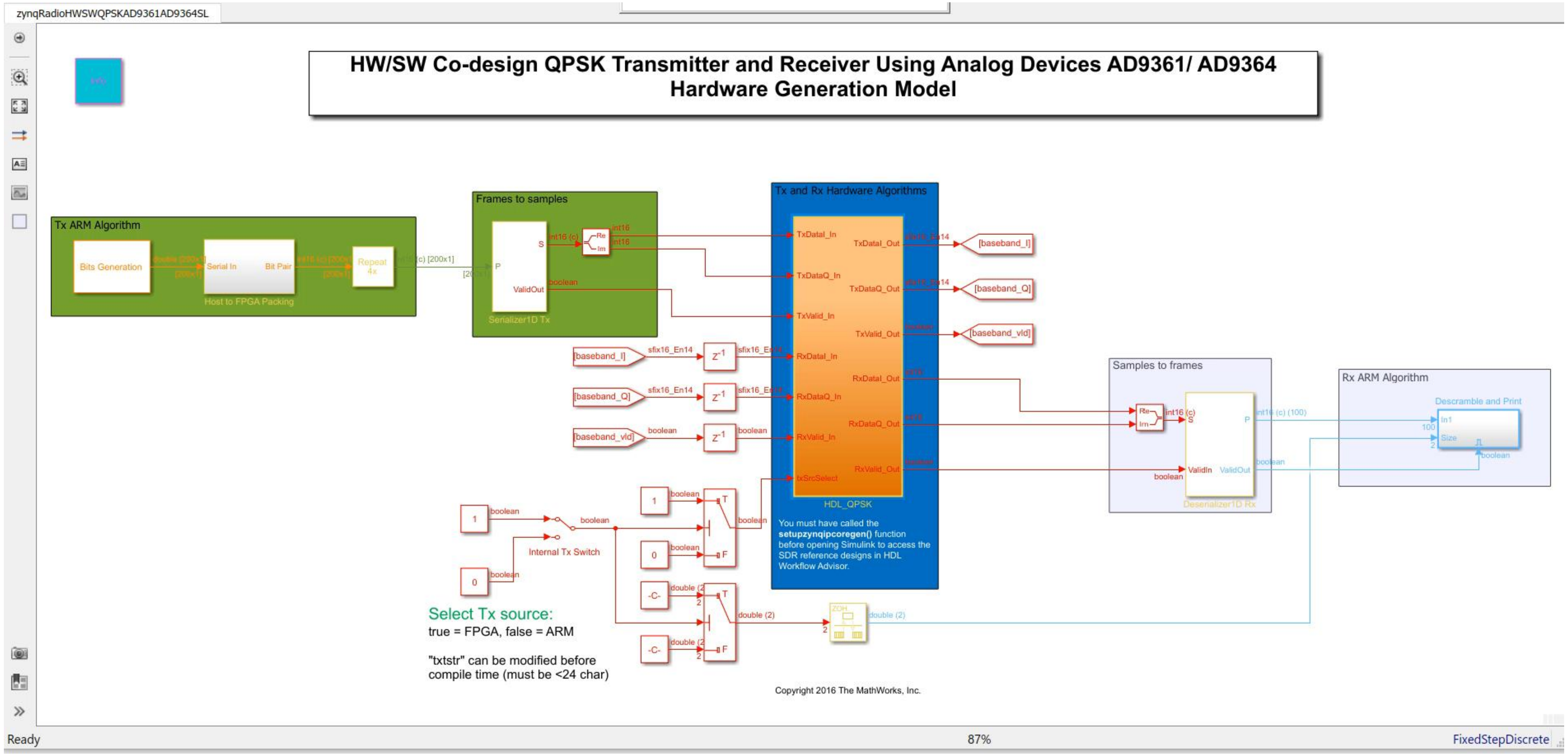




HDL Coder and Embedded Coder supports

- AXI4 Stream Master/Slave
- AXI4 Lite
- AXI4 Stream Master/Slave Video
- AXI4

Interface Configuration



zynqRadioHWSWQPSKAD9361AD9364SL

HW/SW Co-design QPSK Transmitter and Receiver Using Analog Devices AD9361/ AD9364 Hardware Generation Model

You must have called the **setupzynqipcoregen()** function before opening Simulink to access the SDR reference designs in HDL Workflow Advisor.

Select Tx source:
true = FPGA, false = ARM

"txtstr" can be modified before compile time (must be <24 char)

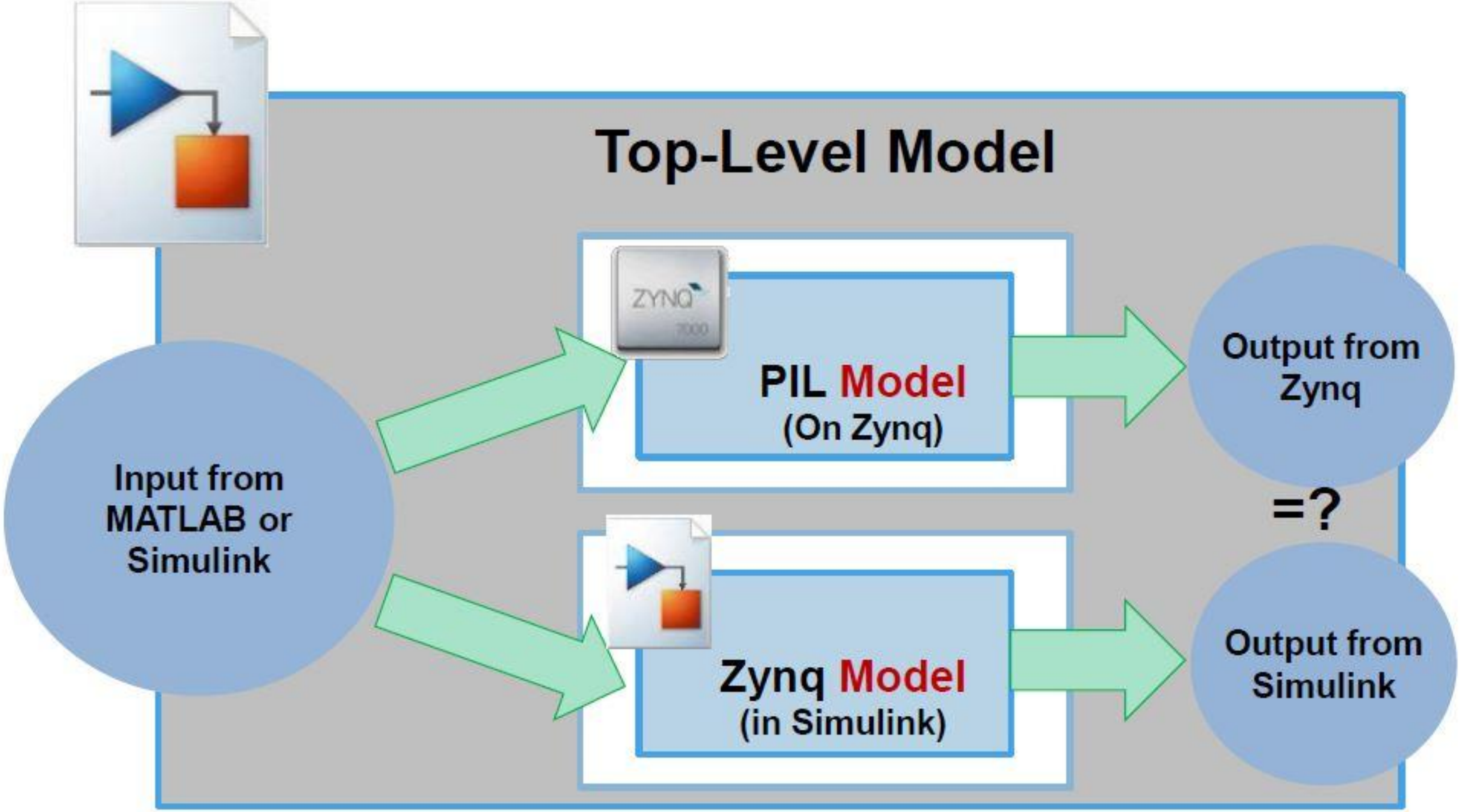
Copyright 2016 The MathWorks, Inc.

Ready 87% FixedStepDiscrete

Addressing Challenges in SoC Design

Partitioning of Algorithm	**Simulink for System Architecture Modeling**
Programming and Verification Expertise	**Automatic C and HDL Code Generation**
Interface Between Software & Hardware	**Generation of AXI Protocol Drivers**
Verification of the Design	• SoC Verification
Applications & Custom Board	• This involves configuring different peripherals

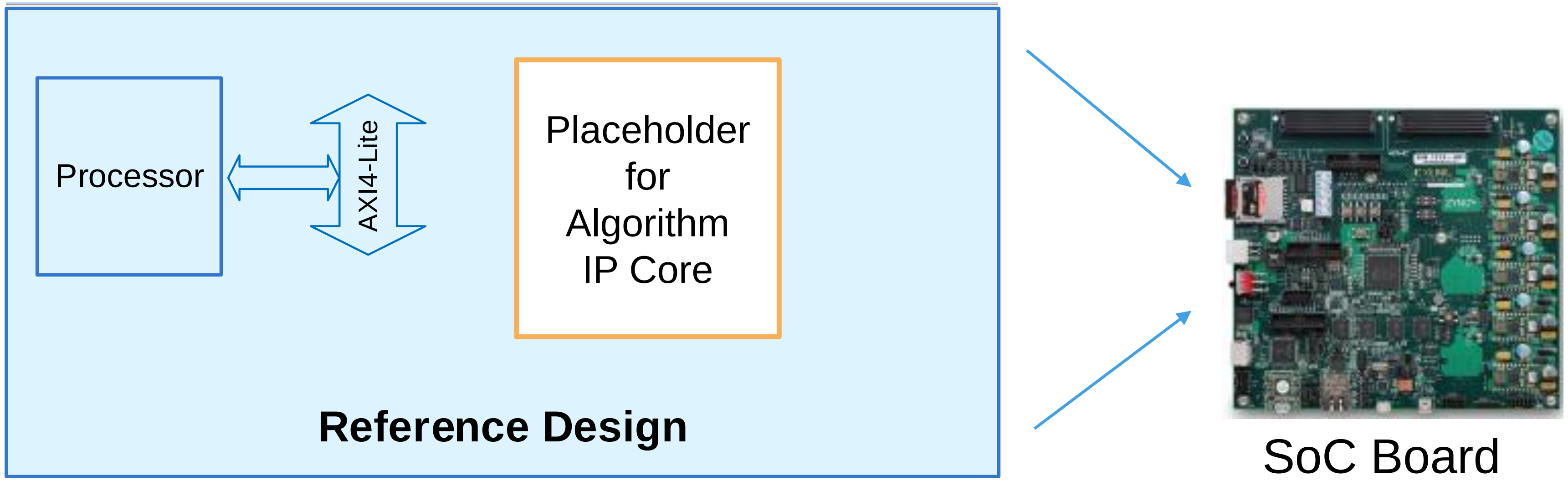
SoC Verification using PIL



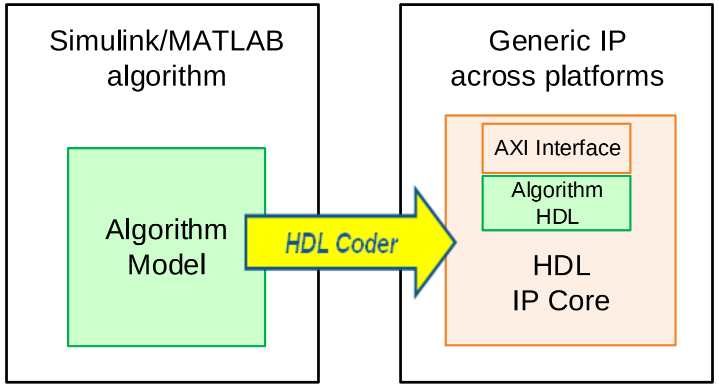
Addressing Challenges in SoC Design

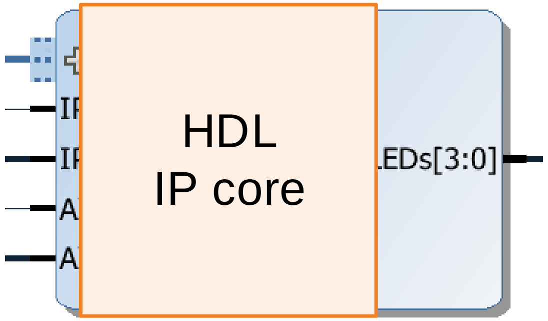
Partitioning of Algorithm	**Simulink for System Architecture Modeling**
Programming and Verification Expertise	**Automatic C and HDL Code Generation**
Interface Between Software & Hardware	**Generation of AXI Protocol Drivers**
Verification of the Design	**Unified Verification Framework**
Applications & Custom Board	• This involves configuring different peripherals

Board and Reference Design

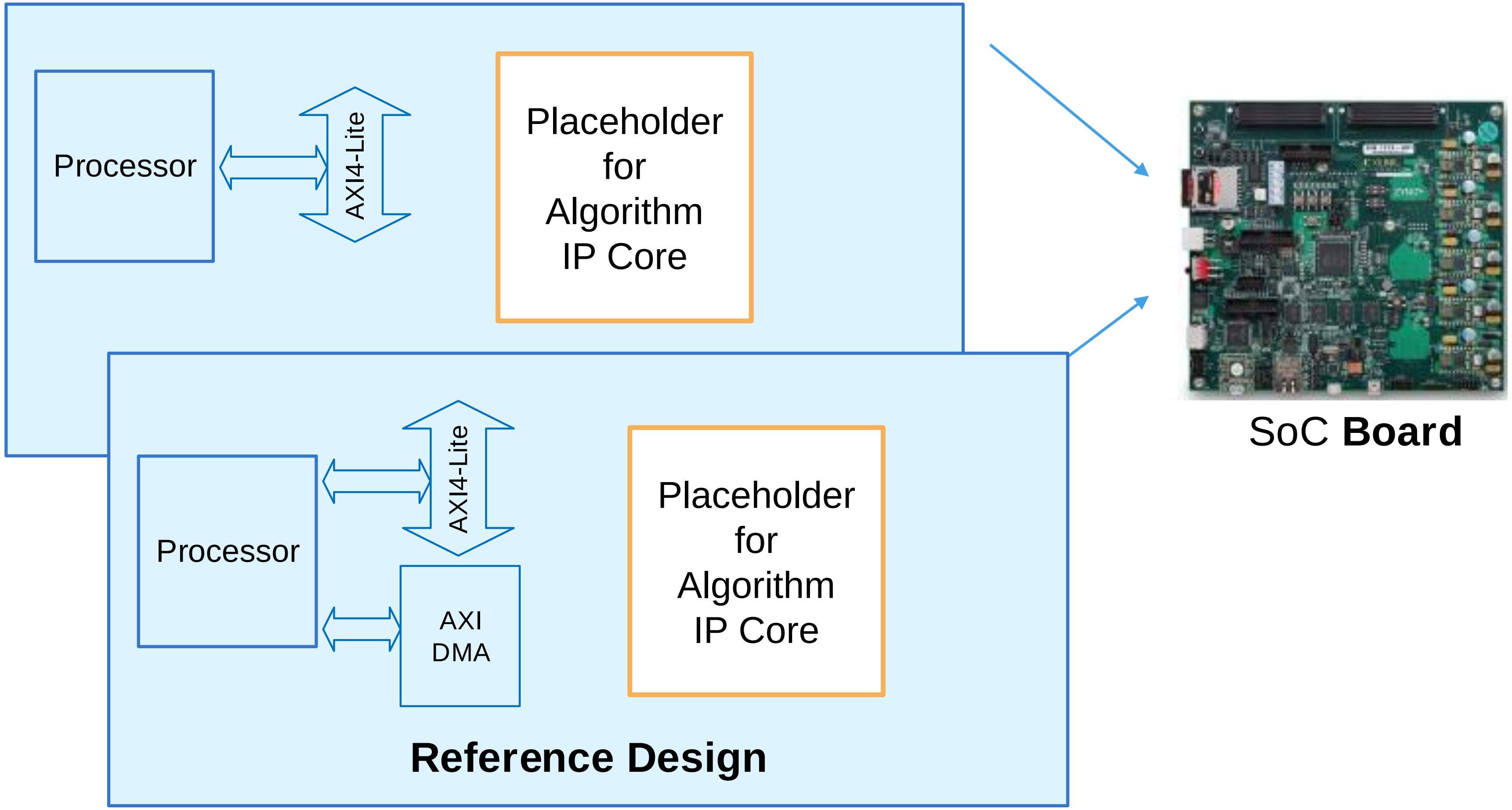


SoC Board
(peripherals,
daughter-cards)

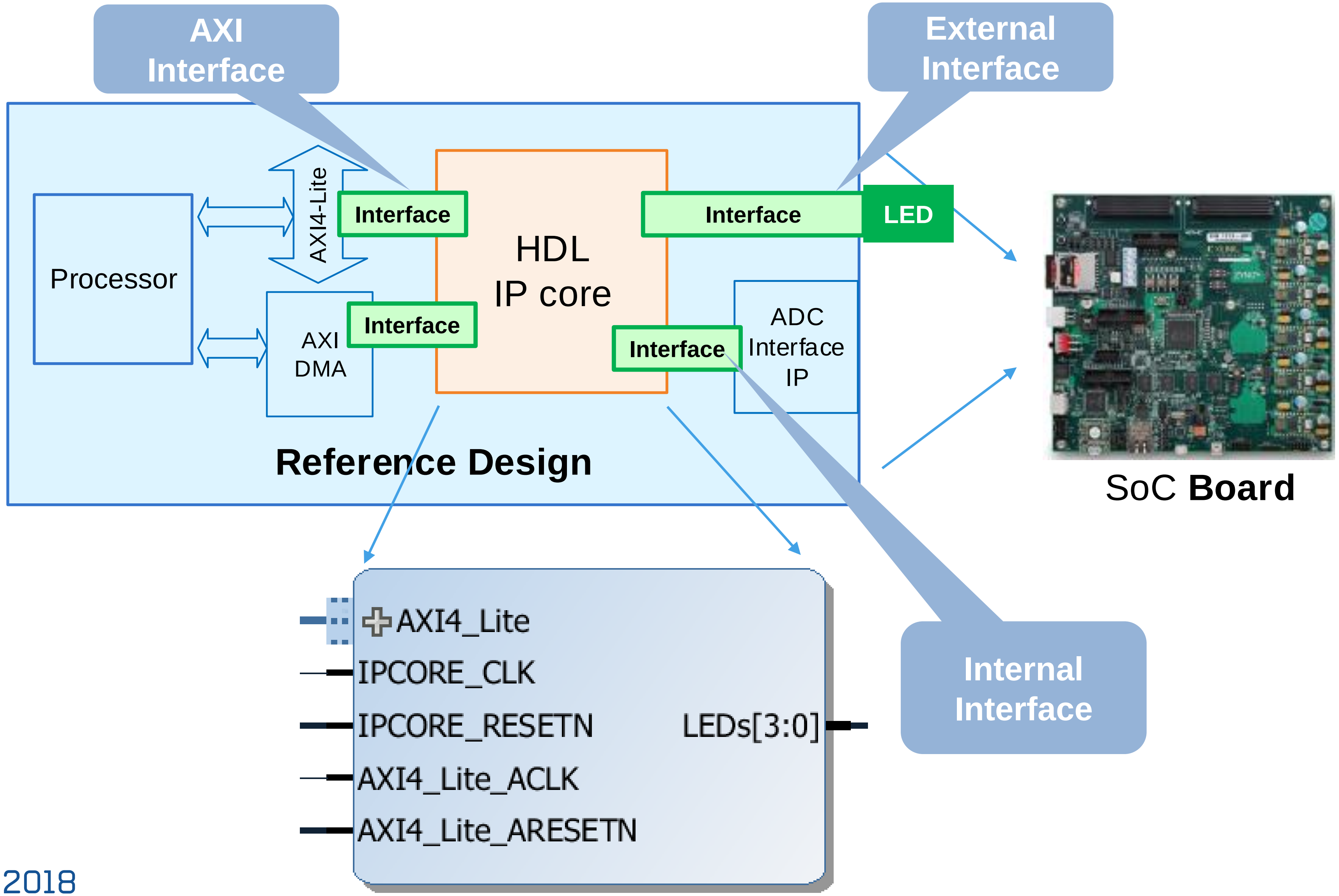




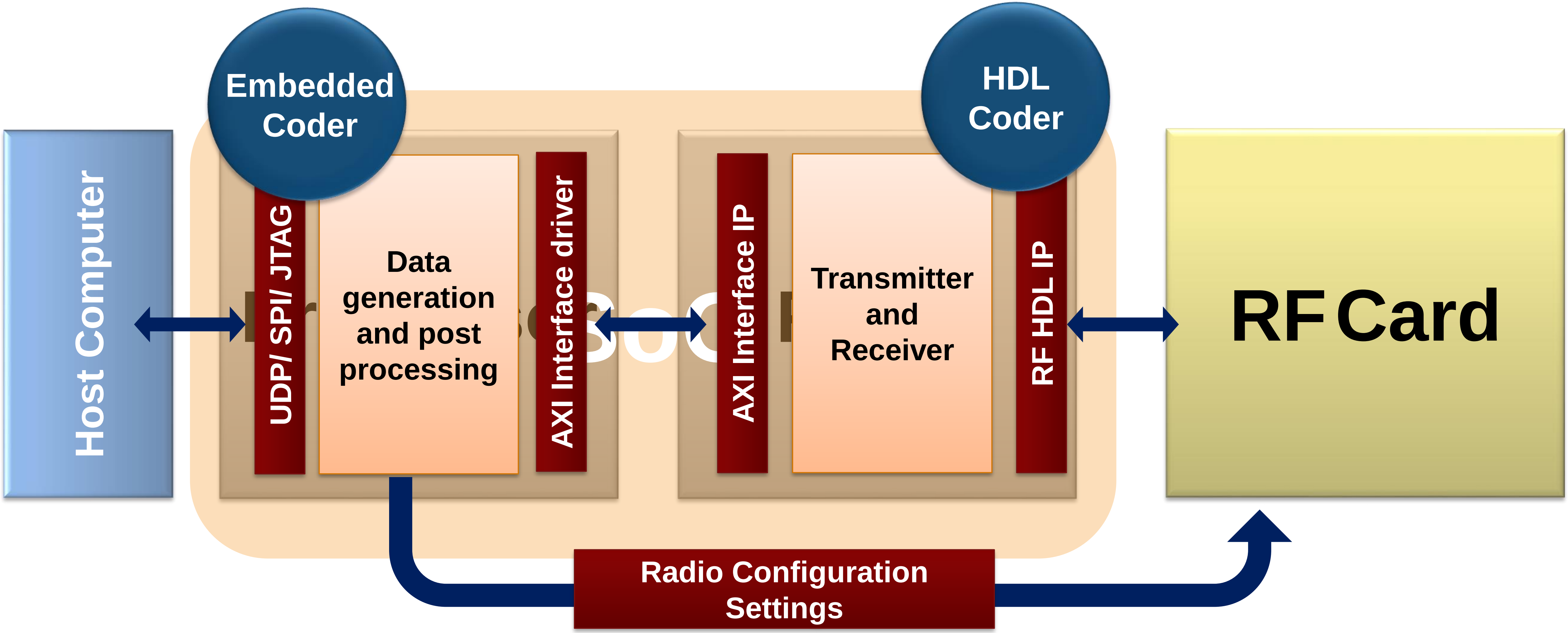
Multiple Reference Designs



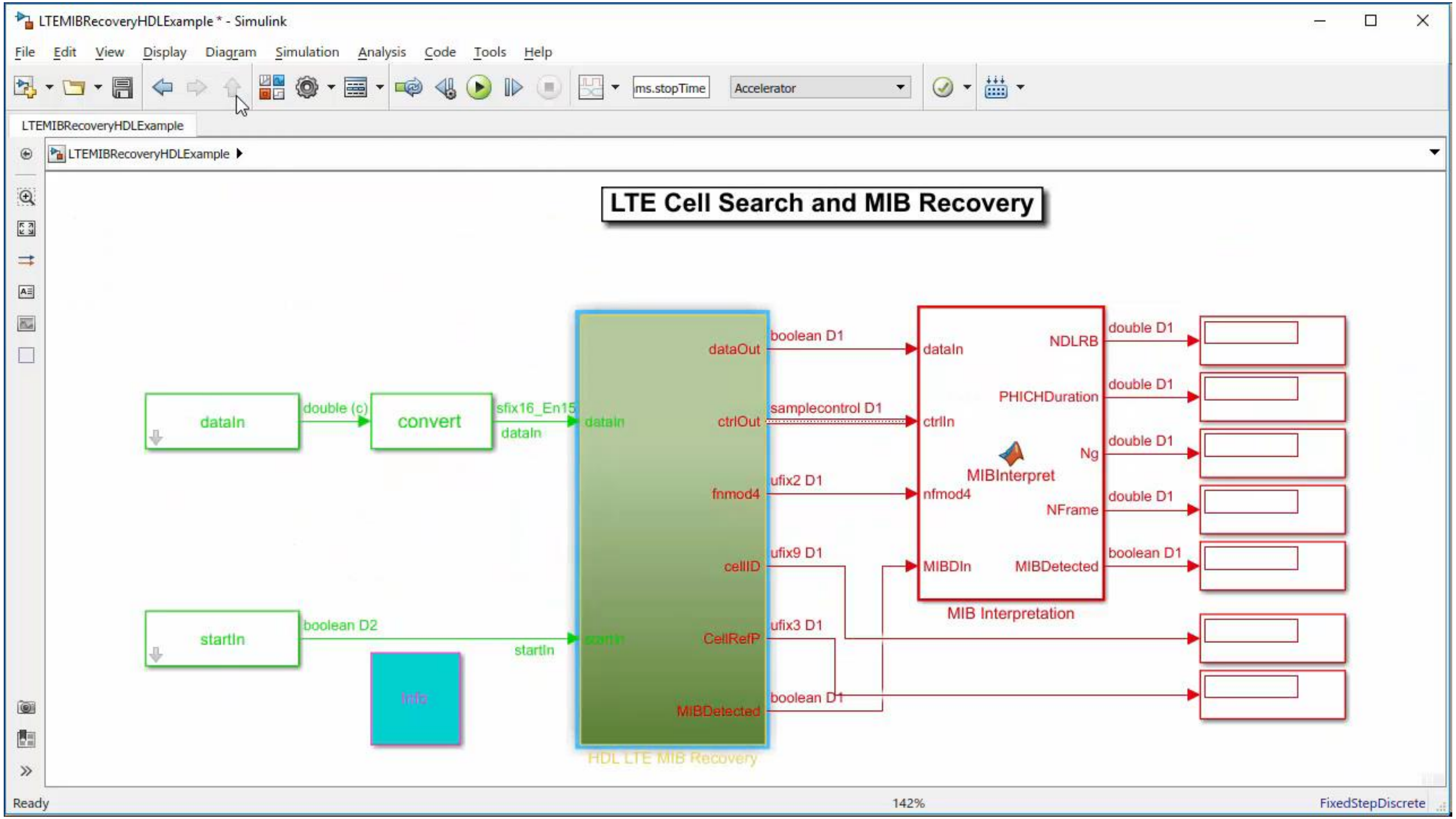
External and Internal Interfaces



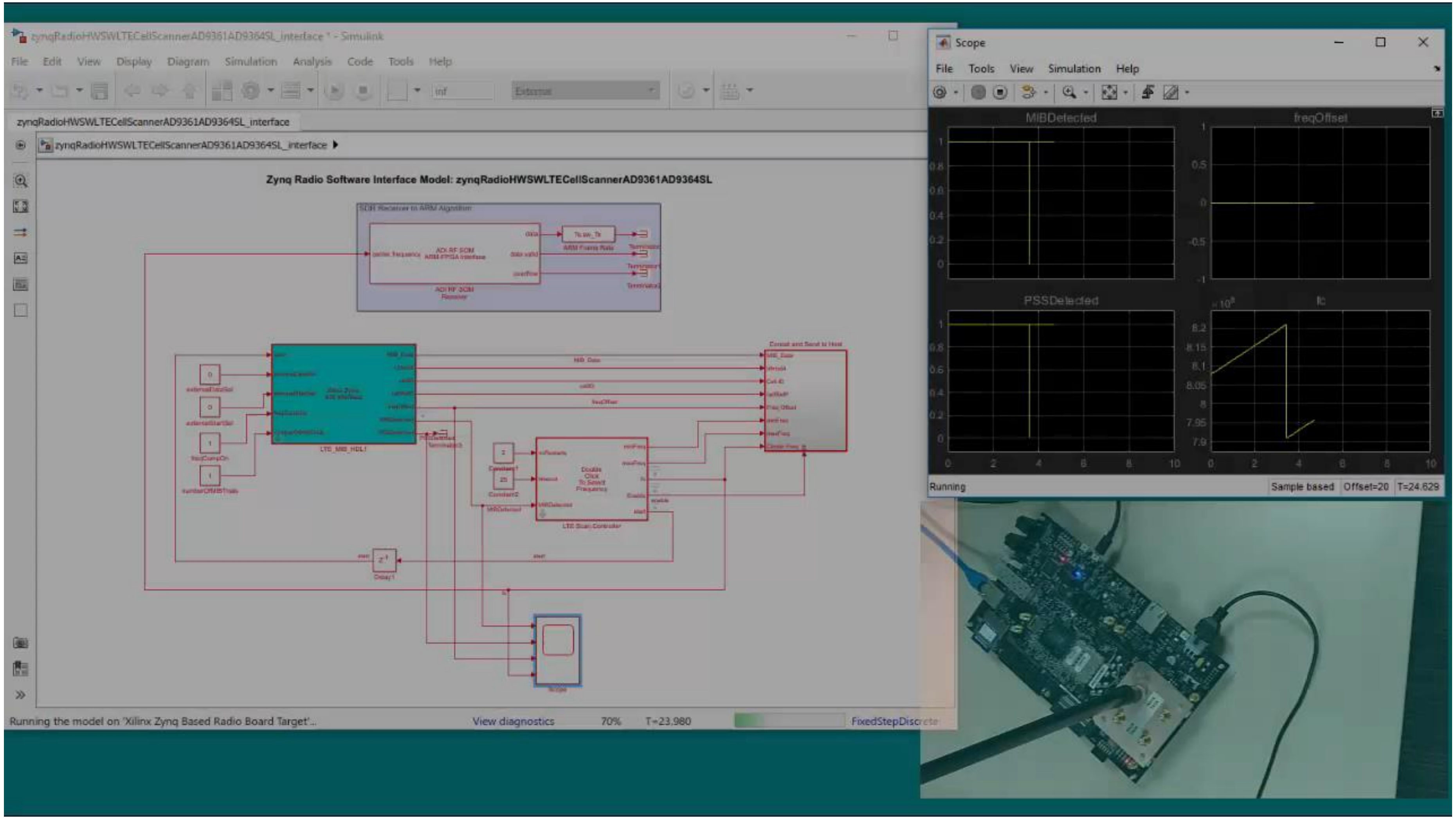
Reference Design for Software Defined Radio



Software Defined Radio Workflow

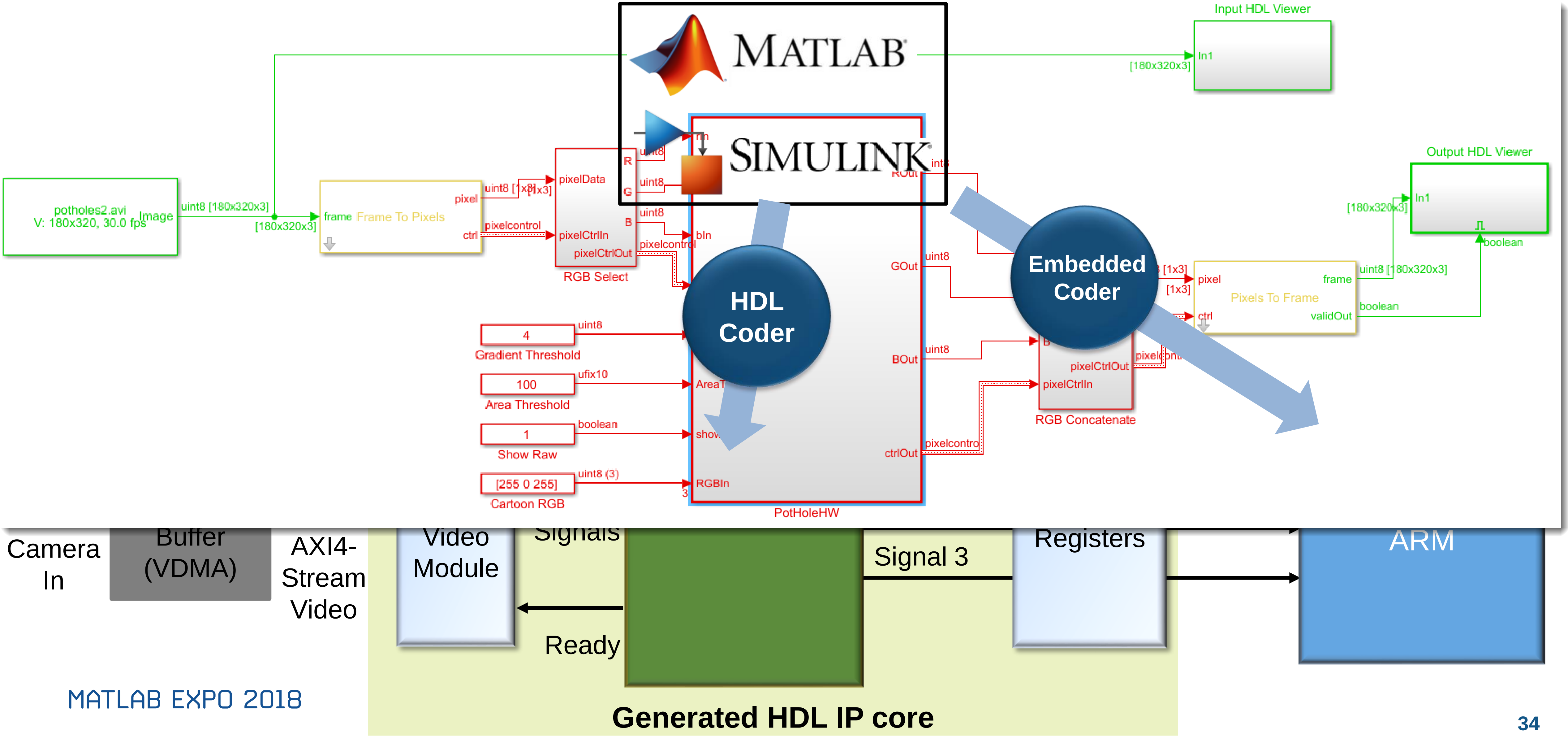


Application Example: Software Defined Radio

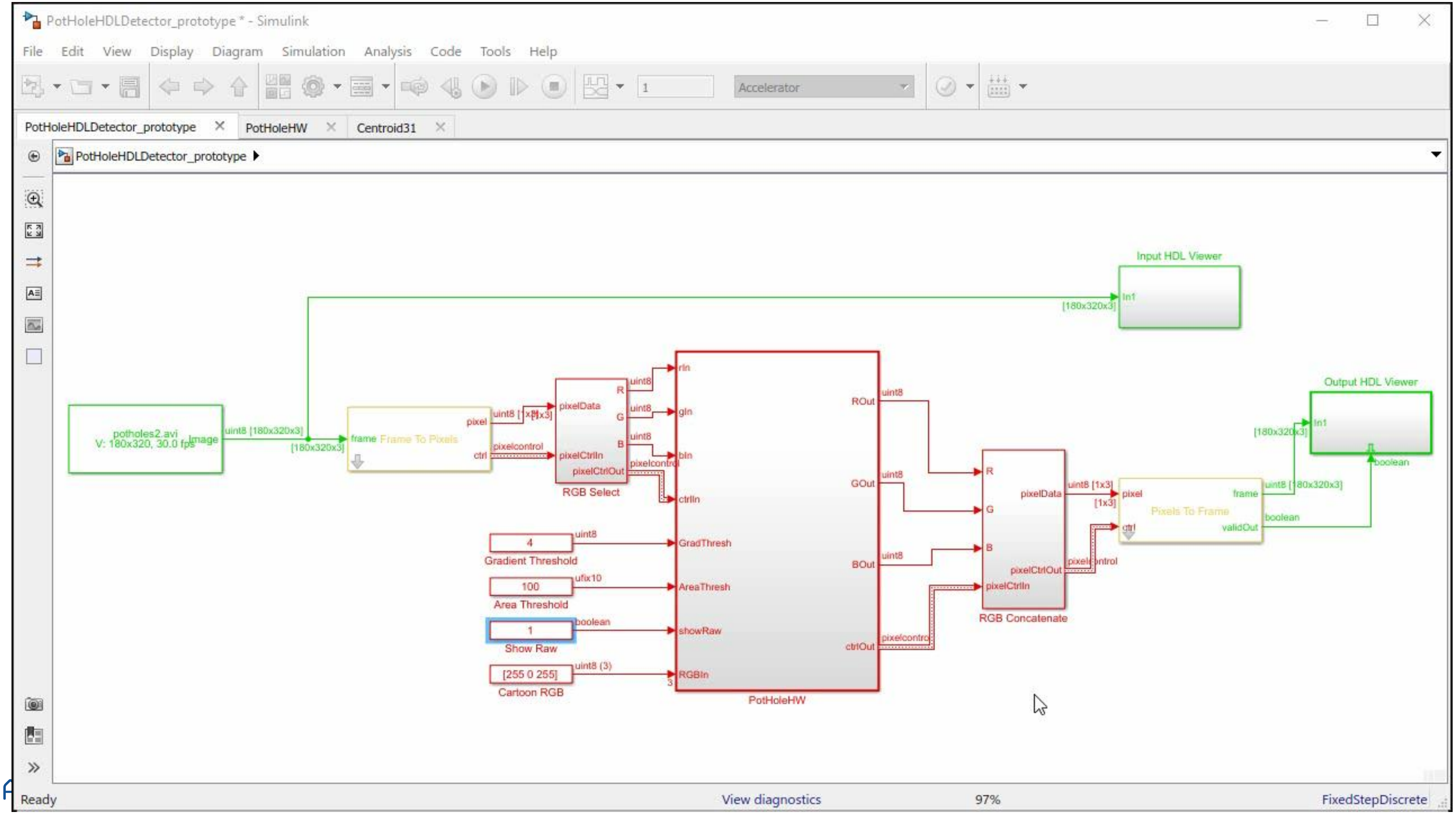


zynqRadioHWSWLTECellScannerAD9361AD9364SL_interface * - Simulink
File Edit View Display Diagram Simulation Analysis Code Tools Help
inf
External
zynqRadioHWSWLTECellScannerAD9361AD9364SL_interface
Zynq Radio Software Interface Model: zynqRadioHWSWLTECellScannerAD9361AD9364SL
SDR Receiver to ARM Algorithm
ADI RF SOM ARM-FPGA interface
ADI RF SOM Receiver
ARM Frame Rate
LTE_MIB_HDL1
LTE Scan Controller
Double Click To Select Frequency
Consat and Send to Host
Delay1
Scope
Running the model on 'Xilinx Zynq Based Radio Board Target'...
View diagnostics
70%
T=23.980
FixedStepDiscrete
Scope
File Tools View Simulation Help
MIBDetected
freqOffset
PSSDetected
fc
Running
Sample based
Offset=20
T=24.629

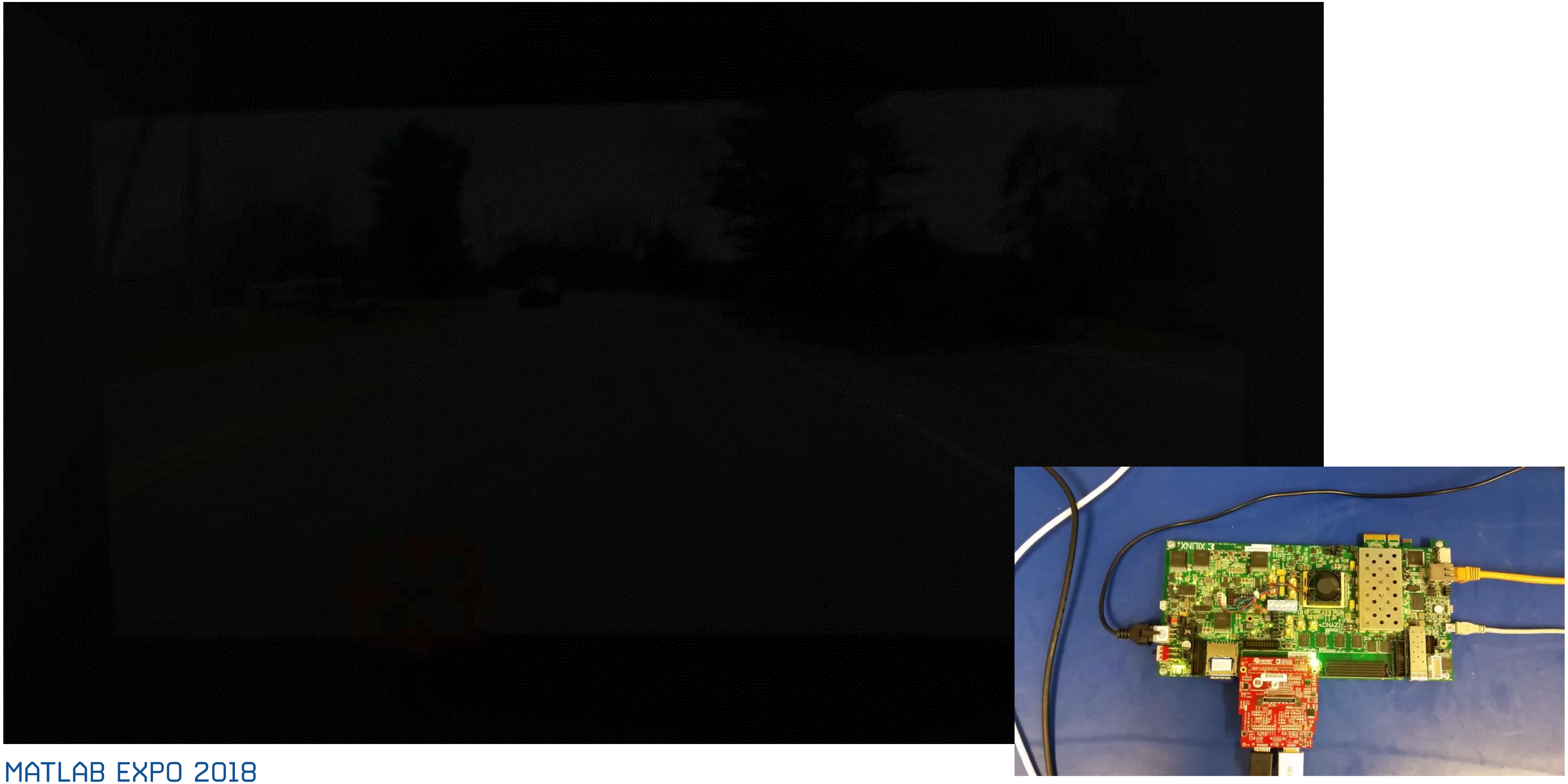
Reference Design for Computer Vision Applications



Application Example: Pot-Hole Detection



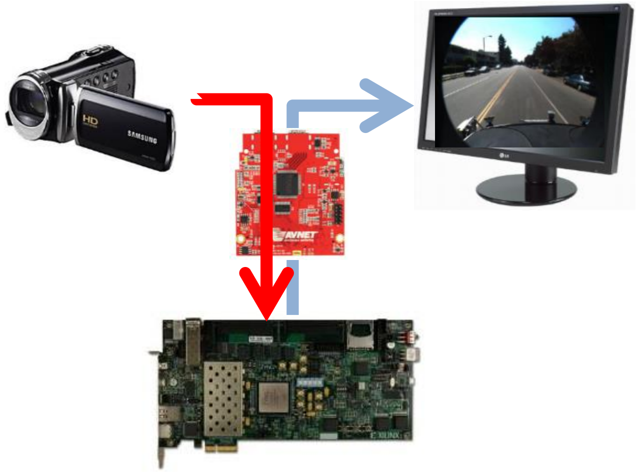
Application Example: Pot-Hole Detection



Supported SoC platforms and Reference Designs

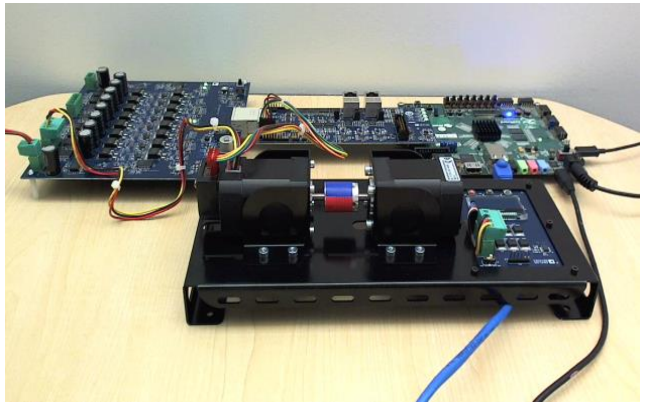
- Xilinx SoCs
  - ZedBoard
  - ZC702 Evaluation Board
  - ZC706 Evaluation Board
  - Analog Devices RF SOM

Video and Image Processing



Software-Defined Radio

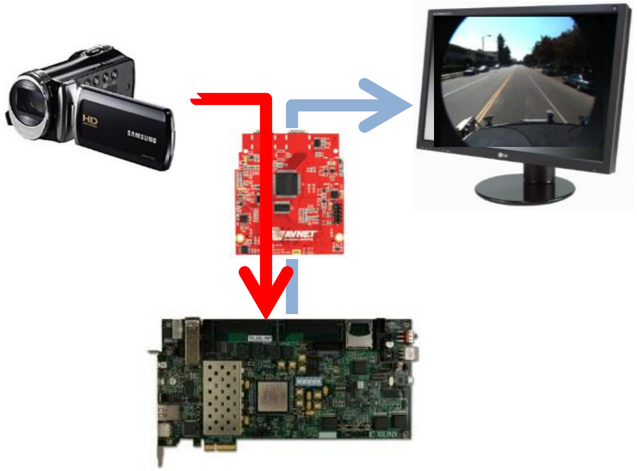
Motor Control



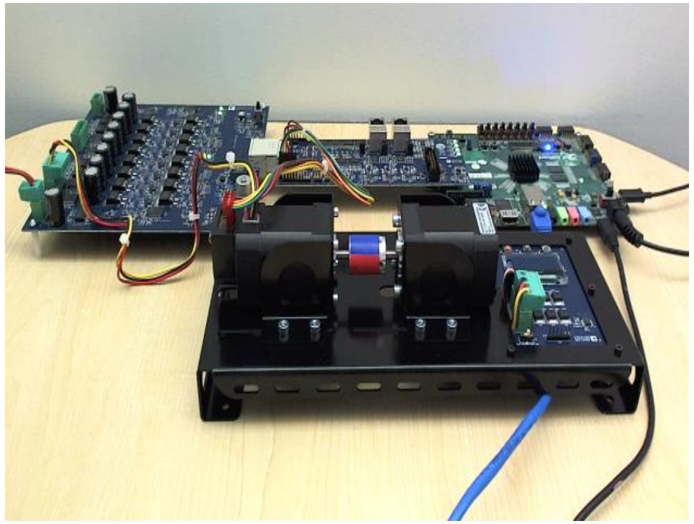
Supported Intel SoC platforms and Reference Designs

- Intel SoCs
  - Arrow SoC
  - Intel Cyclone V SoC

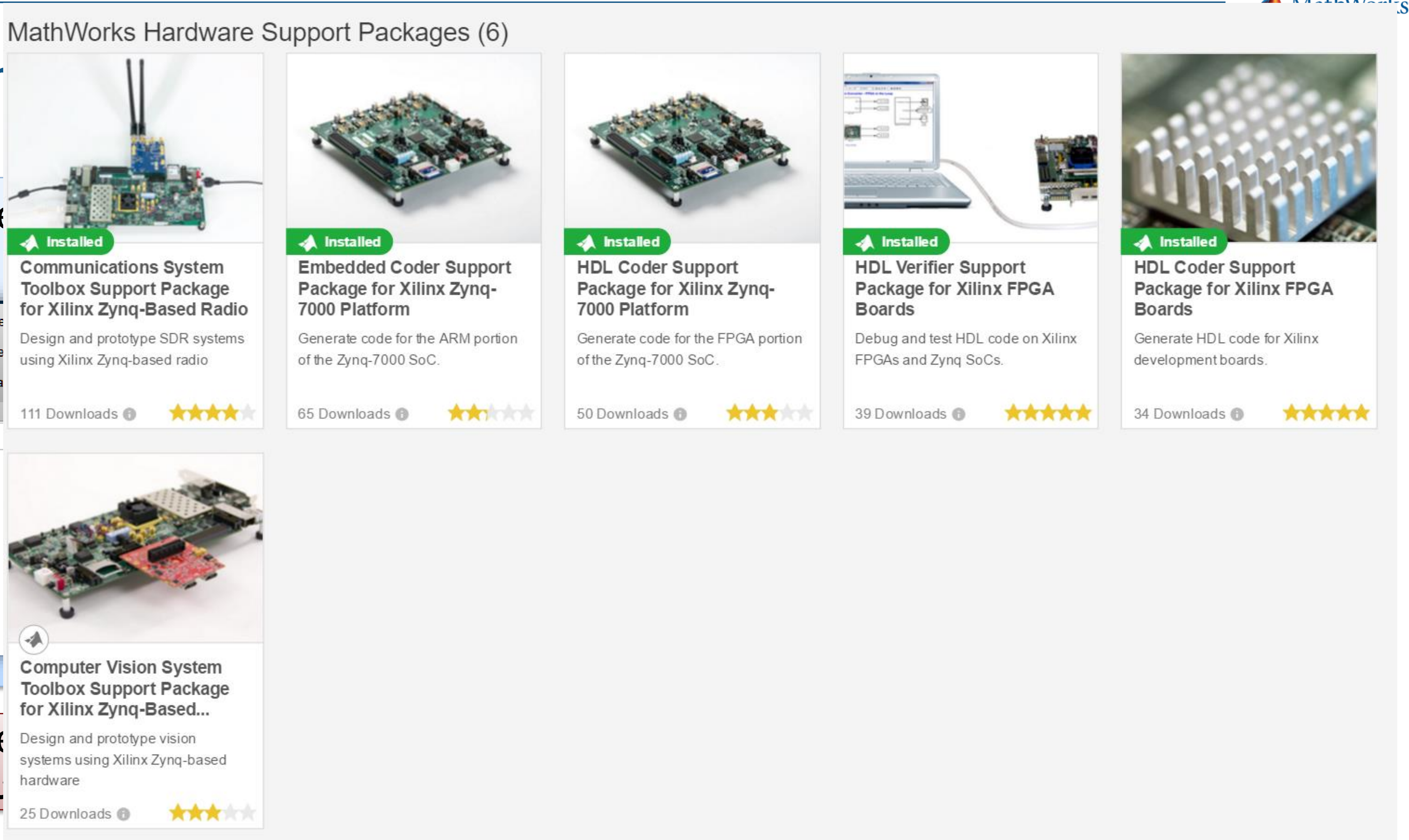
Video and Image Processing



Motor Control



Dowr



MathWorks Hardware Support Packages (6)

Installed
Communications System Toolbox Support Package for Xilinx Zynq-Based Radio
Design and prototype SDR systems using Xilinx Zynq-based radio
111 Downloads ★★★★☆

Installed
Embedded Coder Support Package for Xilinx Zynq-7000 Platform
Generate code for the ARM portion of the Zynq-7000 SoC.
65 Downloads ★★½☆☆

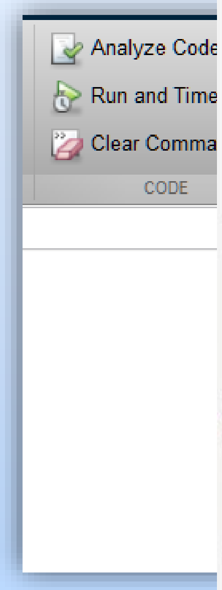
Installed
HDL Coder Support Package for Xilinx Zynq-7000 Platform
Generate code for the FPGA portion of the Zynq-7000 SoC.
50 Downloads ★★★☆☆

Installed
HDL Verifier Support Package for Xilinx FPGA Boards
Debug and test HDL code on Xilinx FPGAs and Zynq SoCs.
39 Downloads ★★★★★

Installed
HDL Coder Support Package for Xilinx FPGA Boards
Generate HDL code for Xilinx development boards.
34 Downloads ★★★★★

Computer Vision System Toolbox Support Package for Xilinx Zynq-Based...
Design and prototype vision systems using Xilinx Zynq-based hardware
25 Downloads ★★★☆☆

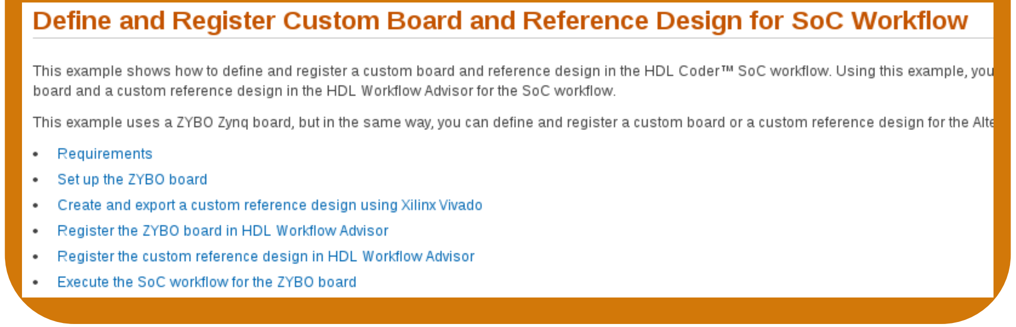
From the
Add-Ons



From the
>> **su**

SoC Custom Reference Design

Example shipping with product

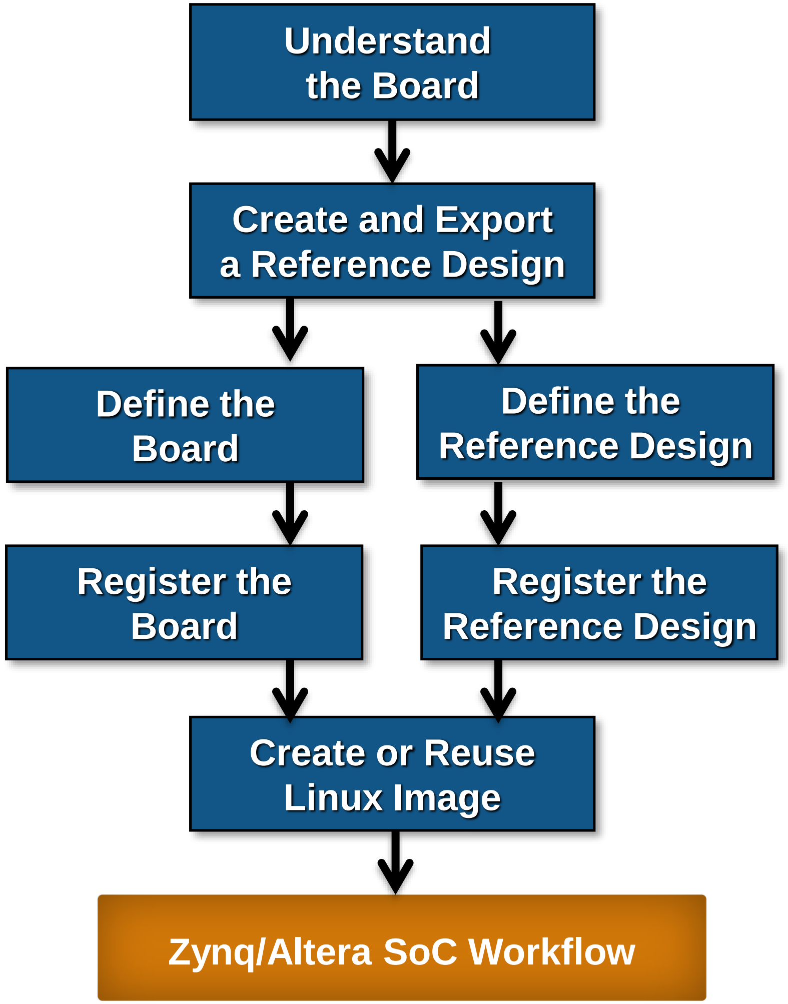


Define and Register Custom Board and Reference Design for SoC Workflow

This example shows how to define and register a custom board and reference design in the HDL Coder™ SoC workflow. Using this example, you board and a custom reference design in the HDL Workflow Advisor for the SoC workflow.

This example uses a ZYBO Zynq board, but in the same way, you can define and register a custom board or a custom reference design for the Alte

- Requirements
- Set up the ZYBO board
- Create and export a custom reference design using Xilinx Vivado
- Register the ZYBO board in HDL Workflow Advisor
- Register the custom reference design in HDL Workflow Advisor
- Execute the SoC workflow for the ZYBO board



Understand the Board

Create and Export a Reference Design

Define the Board

Define the Reference Design

Register the Board

Register the Reference Design

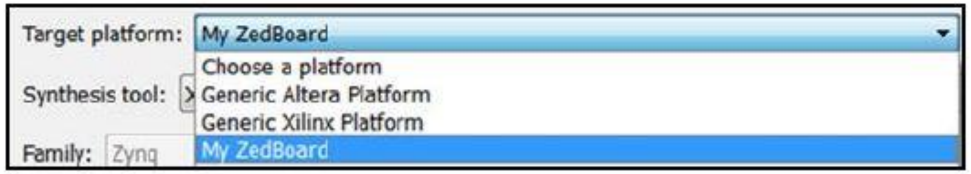
Create or Reuse Linux Image

Zynq/Altera SoC Workflow

https://www.mathworks.com/help/supportpkg/alterasochdlcoder/board-and-reference-design.html

Custom Board Work Flow

```
% Construct reference design object
hRD = hdlcoder.ReferenceDesign('SynthesisTool', 'Xilinx Vivado');
hRD.ReferenceDesignName = 'Default system with OLED and XADC';
hRD.BoardName = 'My ZedBoard';
```

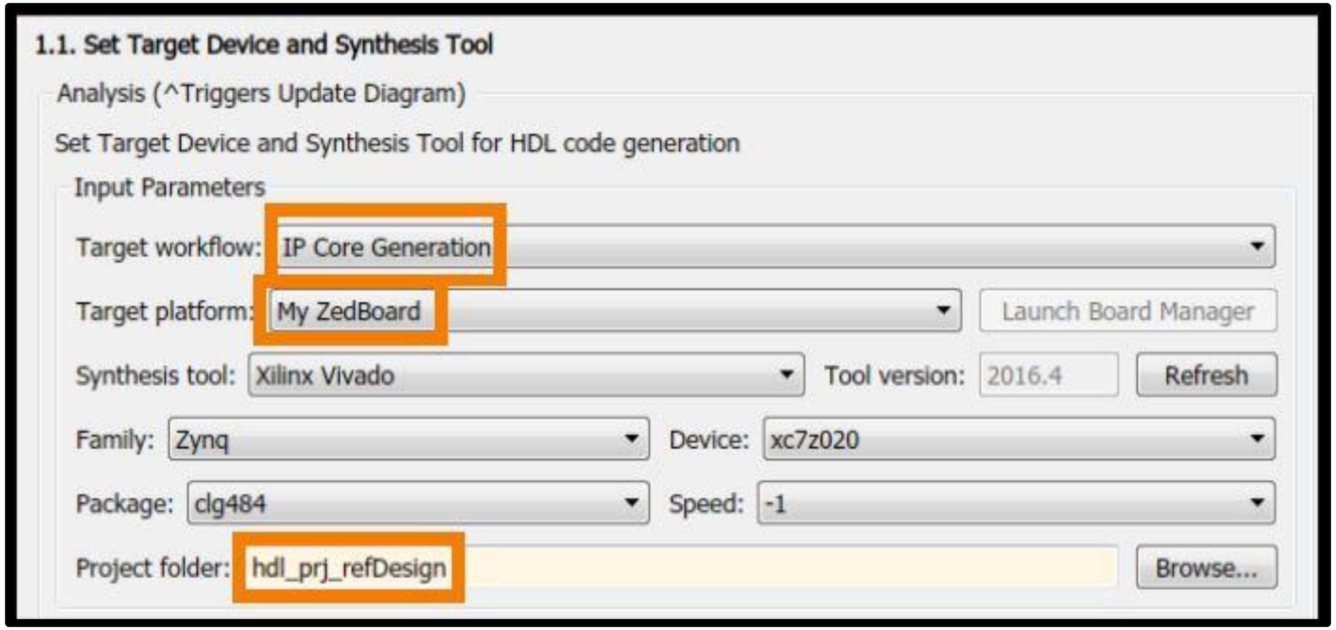


Target platform: My ZedBoard

Choose a platform
Generic Altera Platform
Generic Xilinx Platform
My ZedBoard

Synthesis tool:

Family: Zynq



1.1. Set Target Device and Synthesis Tool

Analysis (^Triggers Update Diagram)

Set Target Device and Synthesis Tool for HDL code generation

Input Parameters

Target workflow: IP Core Generation

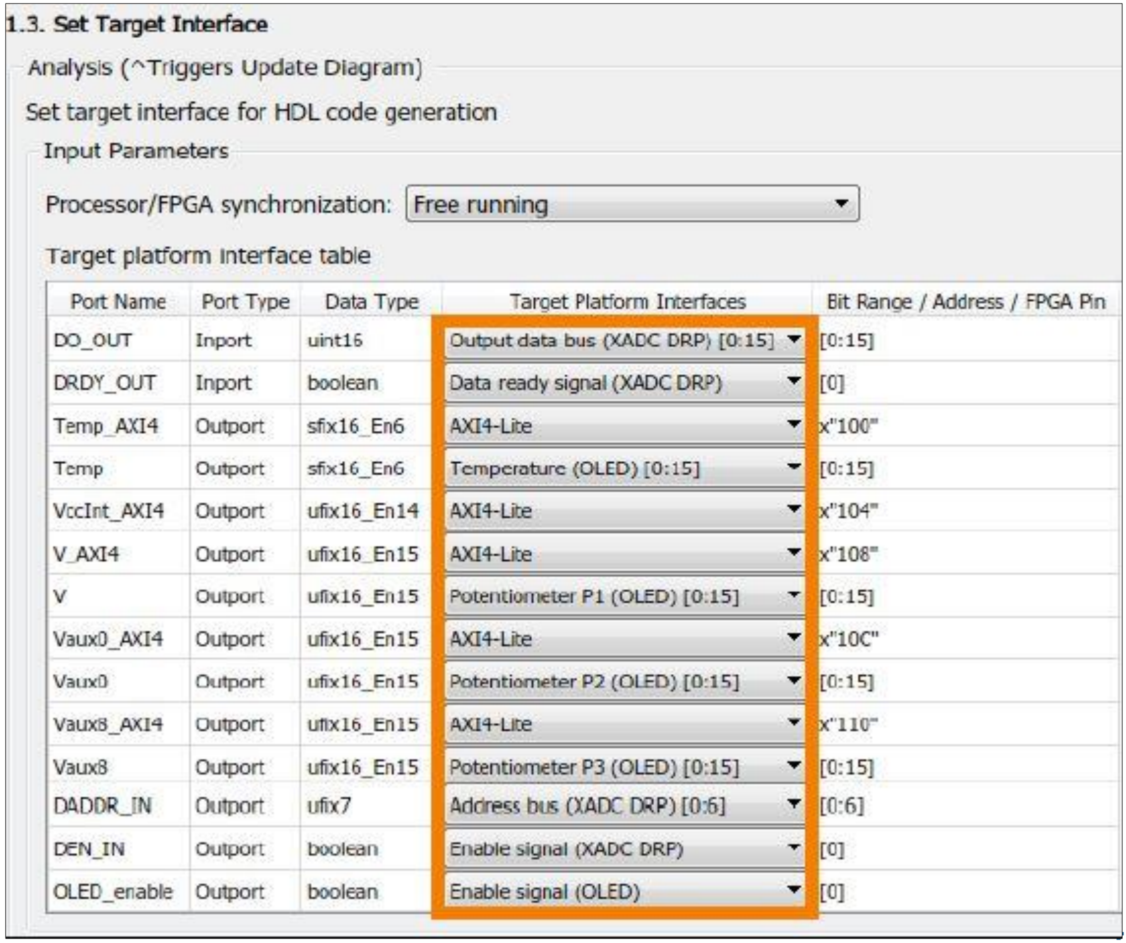
Target platform: My ZedBoard

Synthesis tool: Xilinx Vivado Tool version: 2016.4

Family: Zynq Device: xc7z020

Package: clg484 Speed: -1

Project folder: hdl_prj_refDesign



1.3. Set Target Interface

Analysis (^Triggers Update Diagram)

Set target interface for HDL code generation

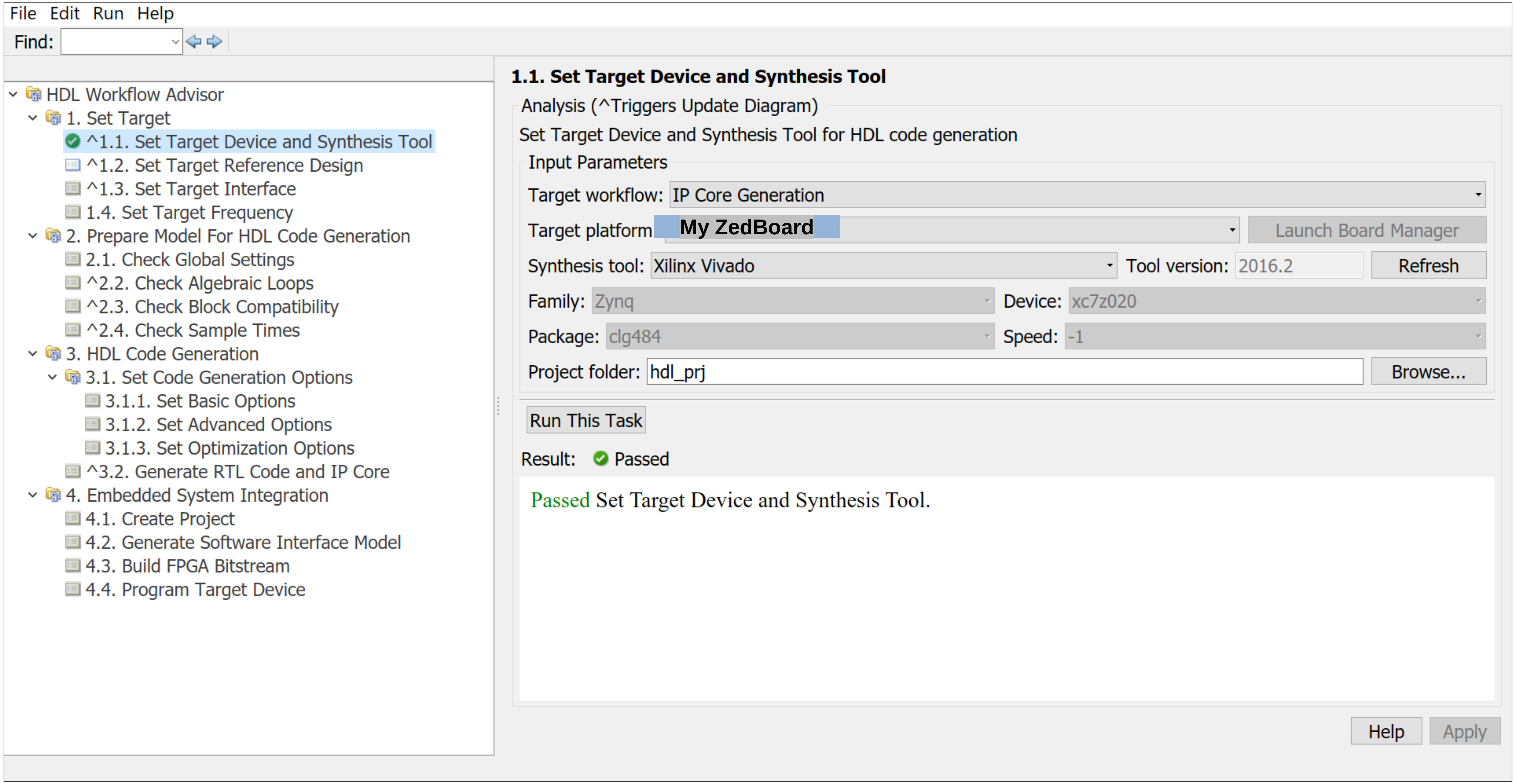
Input Parameters

Processor/FPGA synchronization: Free running

Target platform interface table

Port Name	Port Type	Data Type	Target Platform Interfaces	Bit Range / Address / FPGA Pin
DO_OUT	Inport	uint16	Output data bus (XADC DRP) [0:15]	[0:15]
DRDY_OUT	Inport	boolean	Data ready signal (XADC DRP)	[0]
Temp_AXI4	Outport	sfix16_En6	AXI4-Lite	x"100"
Temp	Outport	sfix16_En6	Temperature (OLED) [0:15]	[0:15]
VccInt_AXI4	Outport	ufix16_En14	AXI4-Lite	x"104"
V_AXI4	Outport	ufix16_En15	AXI4-Lite	x"108"
V	Outport	ufix16_En15	Potentiometer P1 (OLED) [0:15]	[0:15]
Vaux0_AXI4	Outport	ufix16_En15	AXI4-Lite	x"10C"
Vaux0	Outport	ufix16_En15	Potentiometer P2 (OLED) [0:15]	[0:15]
Vaux8_AXI4	Outport	ufix16_En15	AXI4-Lite	x"110"
Vaux8	Outport	ufix16_En15	Potentiometer P3 (OLED) [0:15]	[0:15]
DADDR_IN	Outport	ufix7	Address bus (XADC DRP) [0:6]	[0:6]
DEN_IN	Outport	boolean	Enable signal (XADC DRP)	[0]
OLED_enable	Outport	boolean	Enable signal (OLED)	[0]

Custom Board Work Flow



- HDL Workflow Advisor
  - 1. Set Target
    - ✓ ^1.1. Set Target Device and Synthesis Tool
    - ^1.2. Set Target Reference Design
    - ^1.3. Set Target Interface
    - 1.4. Set Target Frequency
  - 2. Prepare Model For HDL Code Generation
    - 2.1. Check Global Settings
    - ^2.2. Check Algebraic Loops
    - ^2.3. Check Block Compatibility
    - ^2.4. Check Sample Times
  - 3. HDL Code Generation
    - 3.1. Set Code Generation Options
      - 3.1.1. Set Basic Options
      - 3.1.2. Set Advanced Options
      - 3.1.3. Set Optimization Options
    - ^3.2. Generate RTL Code and IP Core
  - 4. Embedded System Integration
    - 4.1. Create Project
    - 4.2. Generate Software Interface Model
    - 4.3. Build FPGA Bitstream
    - 4.4. Program Target Device

1.1. Set Target Device and Synthesis Tool

Analysis (^Triggers Update Diagram)

Set Target Device and Synthesis Tool for HDL code generation

Input Parameters

Target workflow: IP Core Generation

Target platform **My ZedBoard**

Synthesis tool: Xilinx Vivado Tool version: 2016.2

Family: Zynq Device: xc7z020

Package: clg484 Speed: -1

Project folder: hdl_prj

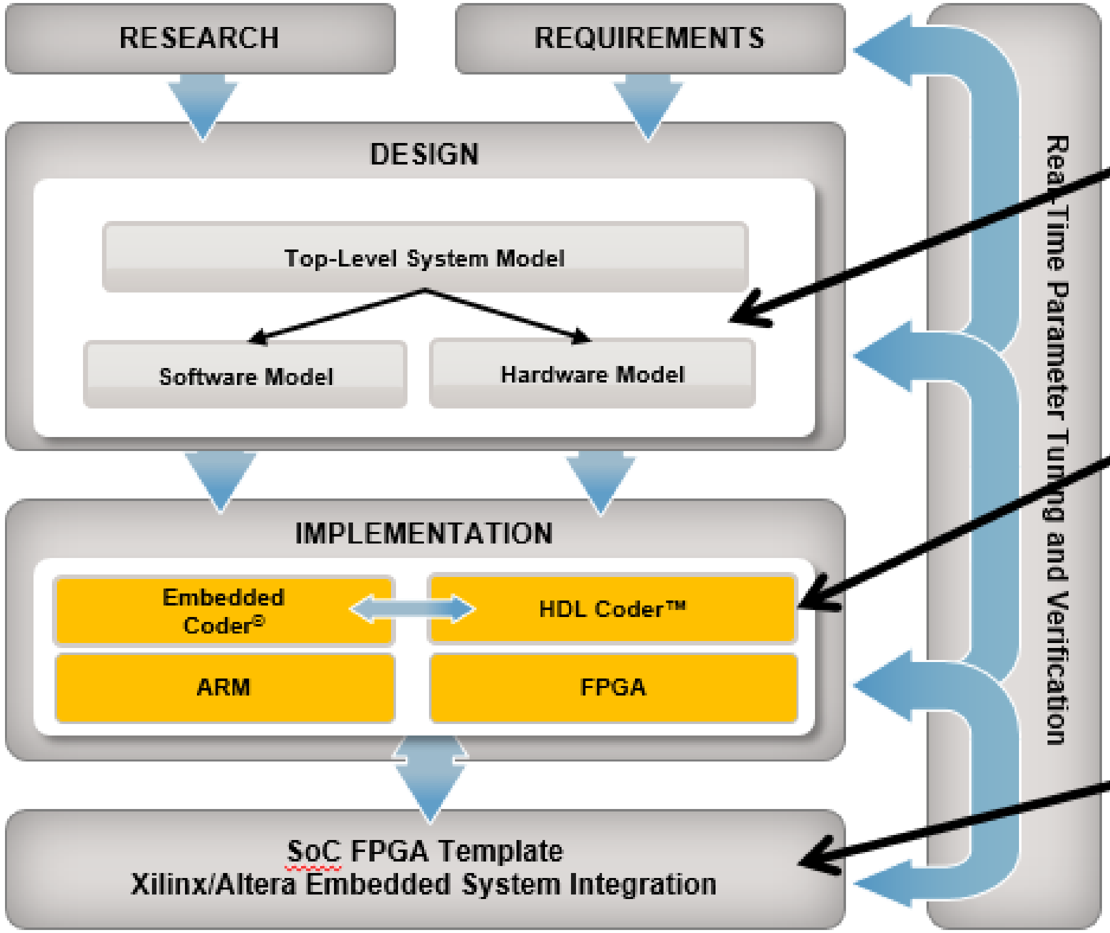
Result: ✓ Passed

Passed Set Target Device and Synthesis Tool.

Addressing Challenges in SoC Design

Partitioning of Algorithm	**Simulink for System Architecture Modeling**
Programming and Verification Expertise	**Automatic C and HDL Code Generation**
Interface Between Software & Hardware	**Generation of AXI Protocol Drivers**
Verification of the Design	**Unified Verification Framework**
Applications & Custom Board	**Reference Designs**

SoC FPGA Design Flow



User defines partitioning

MathWorks automates code and **interface-model generation**

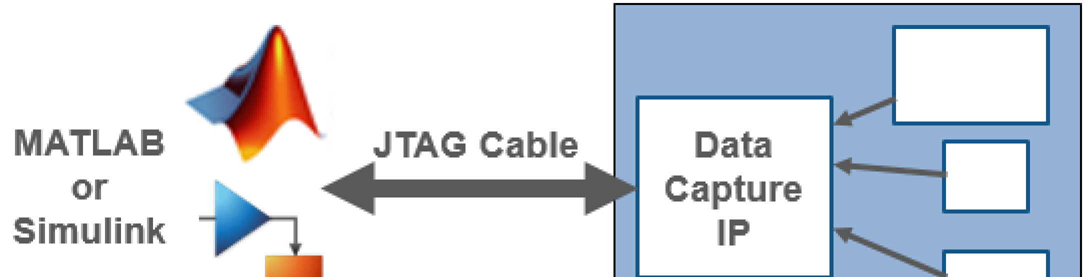
MathWorks automates the **build and download** through the FPGA tools

What’s New in FPGA and SoC Design?

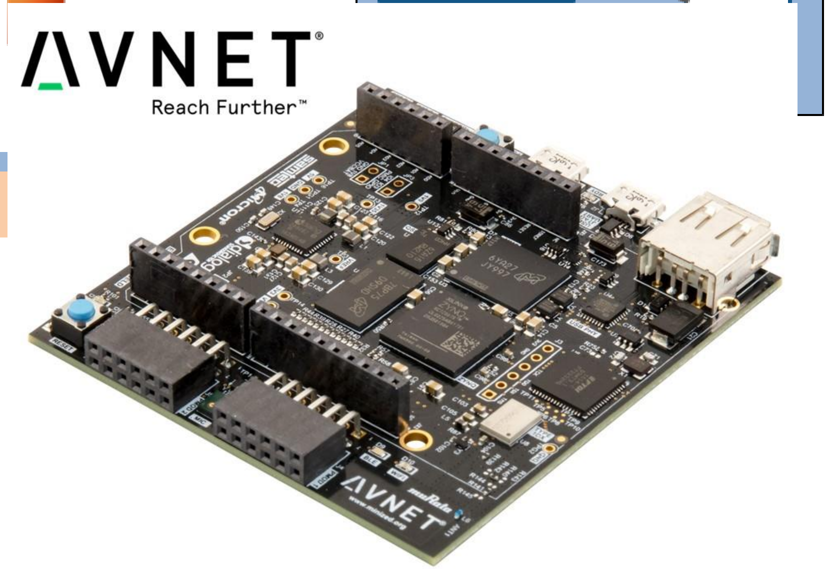
- Floating Point Support R2016b
- FPGA Data Capture R2017a
- MATLAB as AXI Master R2017a
  - Access FPGA External Memory
- HDL Coder Support for Avnet Minize

HDL Coder Native Floating Point

- IEEE-754 Single precision support
- Extensive math and trigonometric operator support



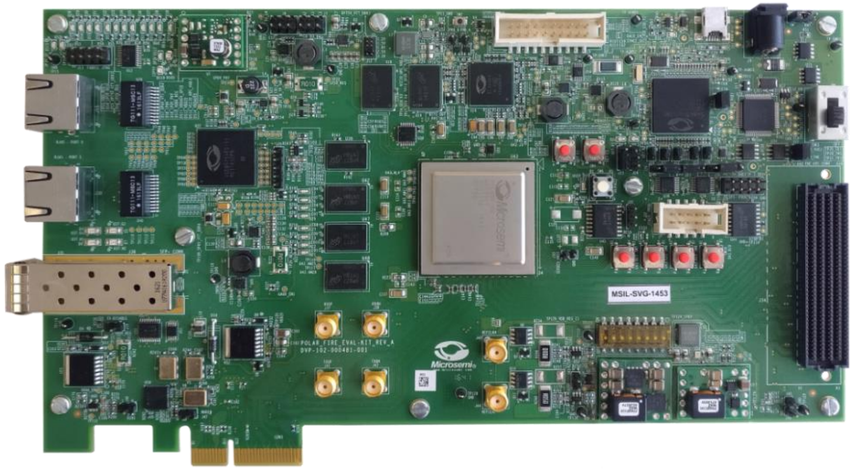
R2018a



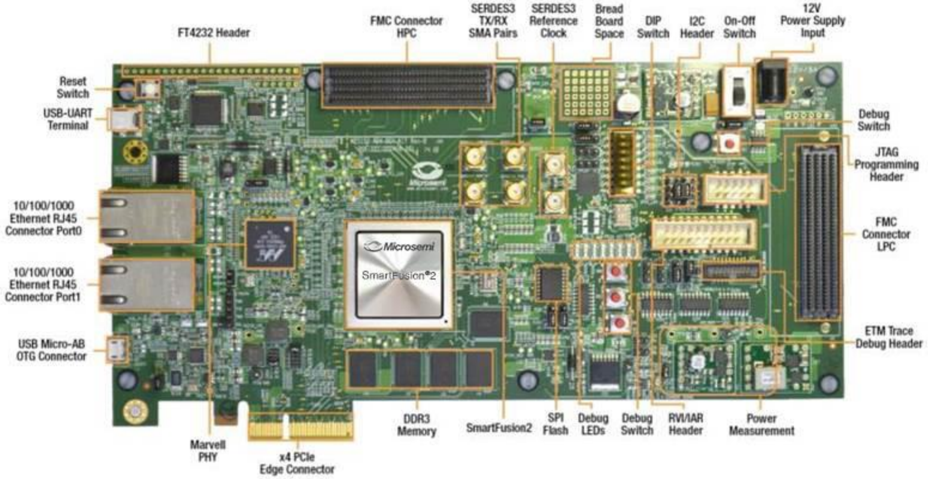
R2017b

FIL Verification: Supported Development Kits

- PolarFire Eval Kit
  - High Performance Kit for full development & testing



- SmartFusion2 Advance Development Kit
  - Full Feature Kit with Advanced Peripherals



Purpose Built Kits for Evaluation and Development of Performance Oriented Low Power Applications

Training Services

Exploit the full potential of MathWorks products

Flexible delivery options:

- Public training available in several cities
- Onsite training with standard or customized courses
- Web-based training with live, interactive instructor-led courses

More than 48 course offerings:

- Introductory and intermediate training on MATLAB, Simulink, Stateflow, code generation, and Polyspace products
- Specialized courses in control design, signal processing, parallel computing, code generation, communications, financial analysis,
  and other areas

www.mathworks.in/training

MathWorks® | *Training Services*

DSP for FPGAs

This three-day course will review DSP fundamentals from the perspective of implementation within the FPGA fabric. Particular emphasis will be given to highlighting the cost, with respect to both resources and performance, associated with the implementation of various DSP techniques and algorithms.

Topics include:

- Introduction to FPGA hardware and technology for DSP applications
- DSP fixed-point arithmetic
- Signal flow graph techniques
- HDL code generation for FPGAs
- Fast Fourier Transform (FFT) Implementation
- Design and implementation of FIR, IIR and CIC filters
- CORDIC algorithm
- Design and implementation of adaptive algorithms such as LMS and QR algorithm
- Techniques for synchronisation and digital communications timing recovery

MathWorks® | *Training Services*

Generating HDL Code from Simulink

two-day course shows how to generate and verify HDL code from a Simulink® model using HDL Coder™ and HDL Verifier™

Topics include:

- Preparing Simulink models for HDL code generation
- Generating HDL code and testbench for a compatible Simulink model
- Performing speed and area optimizations
- Integrating handwritten code and existing IP
- Verifying generated HDL code using testbench and cosimulation

MathWorks | *Training Services*

Programming Xilinx Zynq SoCs with MATLAB and Simulink

two-day course focuses on developing and configuring models in Simulink® and deploying on Xilinx® Zynq®-7000 All Programmable SoCs. For Simulink users who intend to generate, validate, and deploy embedded code and HDL code for software/hardware codesign using Embedded Coder® and HDL Coder™.

A ZedBoard™ is provided to each attendee for use throughout the course. The board is programmed during the class and is yours to keep after the training.

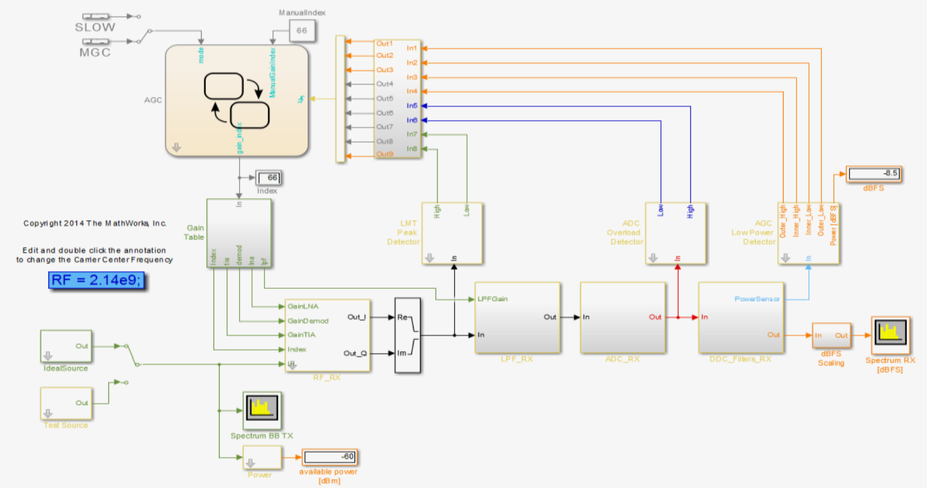
Topics include:

- Zynq platform overview and environment setup, introduction to Embedded Coder and HDL Coder
- IP core generation and deployment, Using AXI4 interface
- Processor-in-the-loop verification, data interface with real-time application
- Integrating device drivers, custom reference design

MathWorks® | *Training Services*

New: Software Defined Radio with Zynq using Simulink

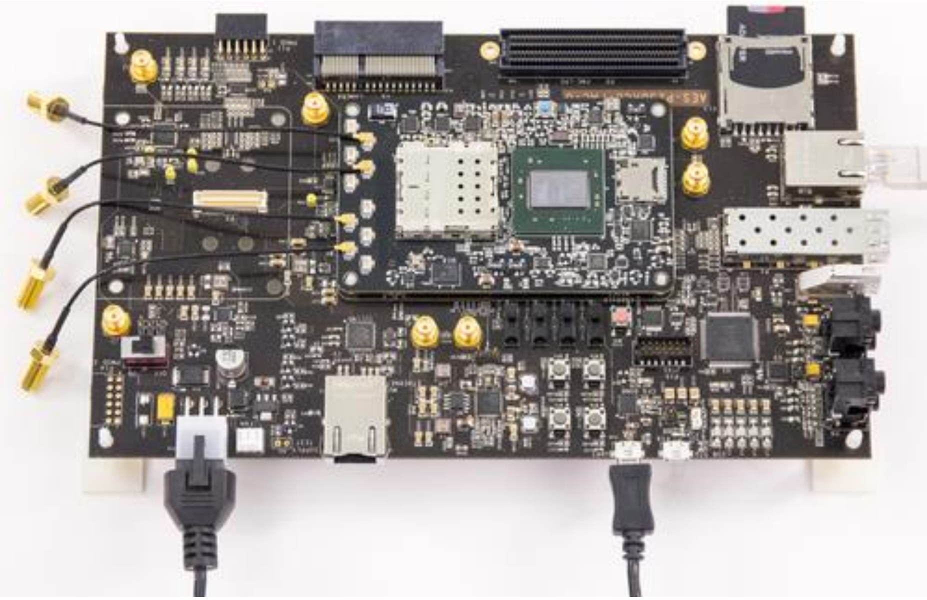
- Learn the Model-Based Design workflow from simulation of RF chain, testing with Radio I/O to moving design to chip
- Get hands-on experience with PicoZed
  - Setting up and communicating with board
  - Capture over-the-air signal and process in MATLAB
  - AD9361 configuration
  - HW/SW co-design for SDR



Copyright 2014 The MathWorks, Inc.

Edit and double click the annotation to change the Carrier Center Frequency

RF = 2.14e9;



MATLAB EXPO 2018

MathWorks Public Training

✓ Guaranteed to run

Upcoming Public Trainings `	Dates	Location
DSP for FPGAs	May 7-9 ✓	Bangalore
DSP for FPGAs	June 11-13	Hyderabad
Generating HDL Code from Simulink	Aug 27 - 28	Hyderabad
Programming Xilinx Zynq SoCs using MATLAB and Simulink	Aug 29 - 30	Hyderabad
Software Defined Radio with Zynq using Simulink	Aug 31	Hyderabad
Generating HDL Code from Simulink	Oct 8 -9 ✓	Bangalore
Programming Xilinx Zynq SoCs using MATLAB and Simulink	Oct 10 - 11 ✓	Bangalore
Software Defined Radio with Zynq using Simulink	Oct 12 ✓	Bangalore

Email: training@mathworks.in URL: http://www.mathworks.in/services/training Phone: 080-6632-6000

Call to Action

Videos and Webinars

- White Paper: Deploying LTE Wireless Communications on FPGAs: A Complete MATLAB and Simulink Workflow
- Webinars:
  - Modeling HDL Components for FPGAs in Control Applications
  - Prototyping SoC-based Motor Controllers with MATLAB and Simulink
  - Radio Deployment on SoC Platforms
- Video Series: Vision Processing for FPGA (5 Videos)
- Upcoming webinar on Microsemi Integration with MATLAB Tools
- Custom Reference Design

MathWorks®

Accelerating the pace of engineering and science

Speaker Details

Email: Hitu.Sharma@mathworks.in
Vinod.Thomas@mathworks.in

Contact MathWorks India

Products/Training Enquiry Booth

Call: 080-6632-6000

Email: info@mathworks.in

Your feedback is valued.

Please complete the feedback form provided to you.

SIL MOdel

Intel SoC support
Training Dates