

MathWorks Tools for Software-Defined Radios, Wireless Prototyping and Verification

Houman Zarrinkoub, PhD.

Product Manager
Signal Processing & Communications

MathWorks
houmanz@mathworks.com

Software Defined Radios (SDR) Today

Important Tool for

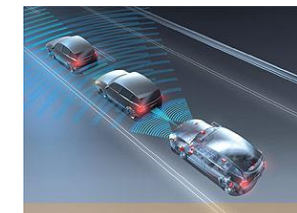
- Education
- Research
- Development

Set to expand in scope and use

- Internet of Things
- Connected devices

Useful in

- rapid prototyping of wireless algorithms
- Evaluate radio hardware in real-world conditions



How to make SDRs ubiquitous for Wireless Prototyping

- Affordable SDR hardware
 - Sufficient computational power
 - Support a range of carrier frequencies & bandwidths
 - Support multiple antennas
- SDR software development environments
 - high level of control of the SDR platform
 - Exchange data to SDR hardware in real time
- Connectivity to powerful technical computing software
 - use reliable software models and tools in experiments
 - algorithms, and feature-rich
 - Use standard waveforms

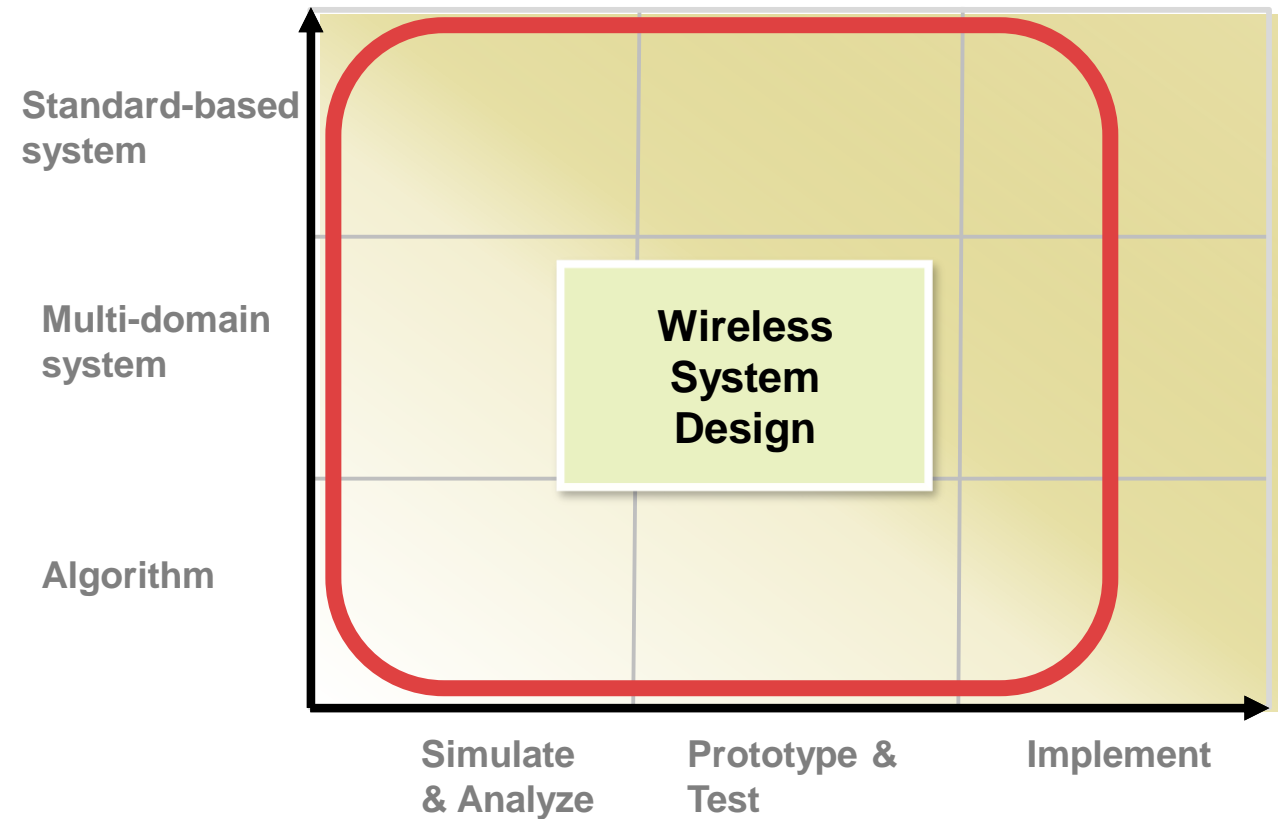
Wireless Prototyping

Who are the users?

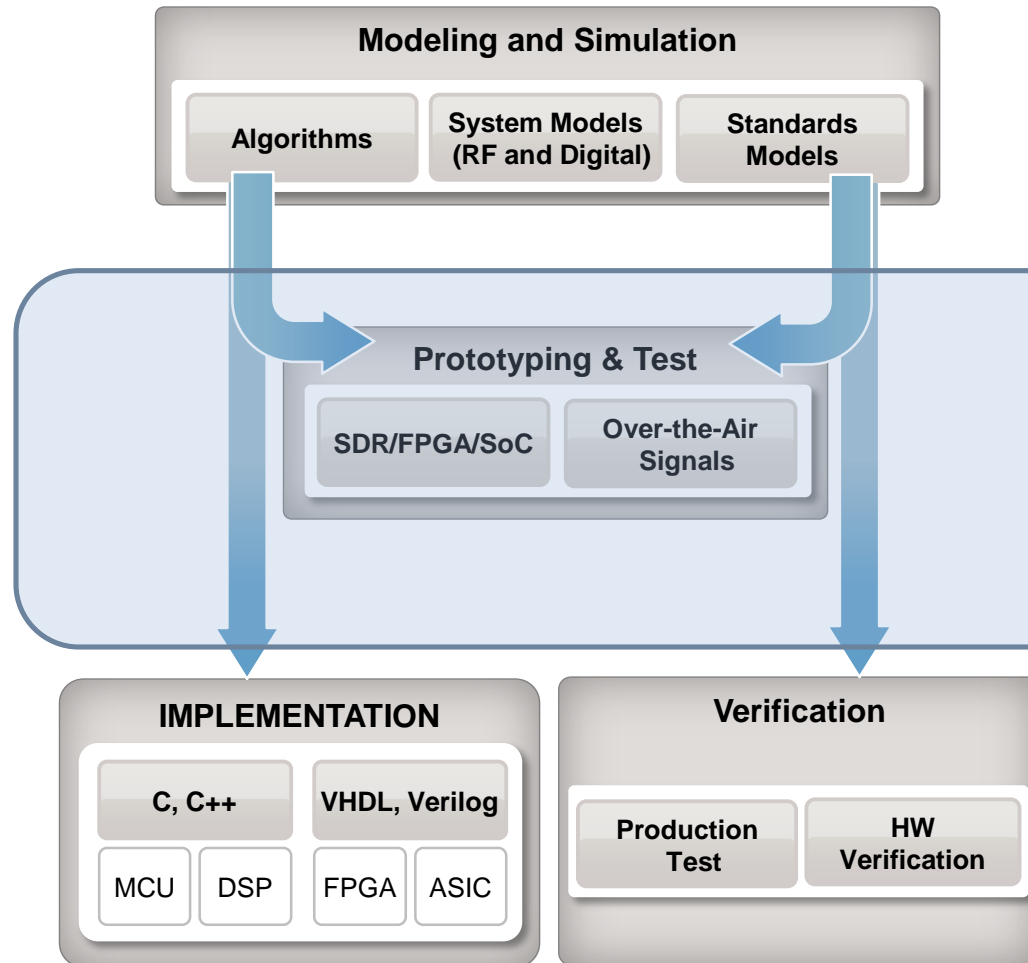
- Communications engineer
- RF System engineer
- Test or validation engineer

What do they need?

- End-to-end simulation
- Design verification
- Real-world over-the-air testing
- Standard-compliant waveforms



Model-Based Design: As Algorithms move toward Implementations



Modeling and Simulation

- Mathematical modeling & algorithm design
- Multi-domain simulation: Digital + RF
- More and more wireless standard models

Prototyping and Testing

- Test with a range of SDR and instrument hardware
- Rapid prototyping with code generation

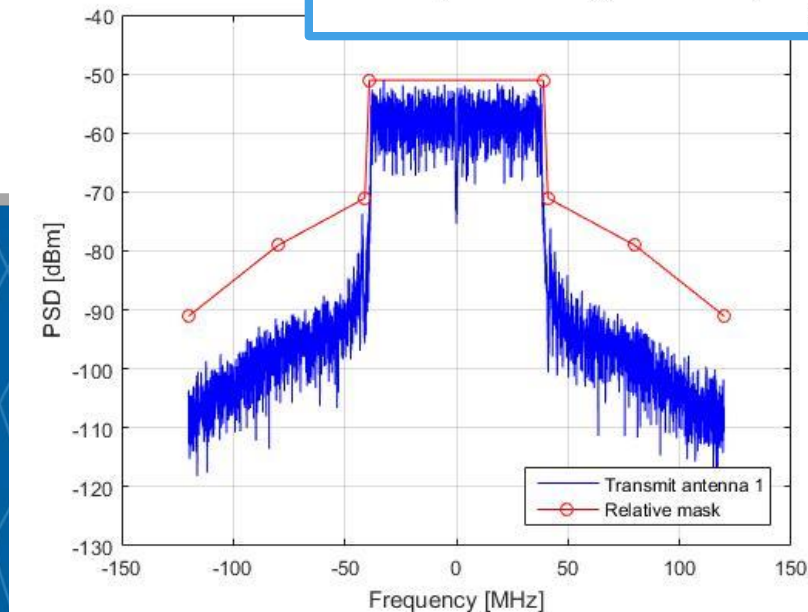
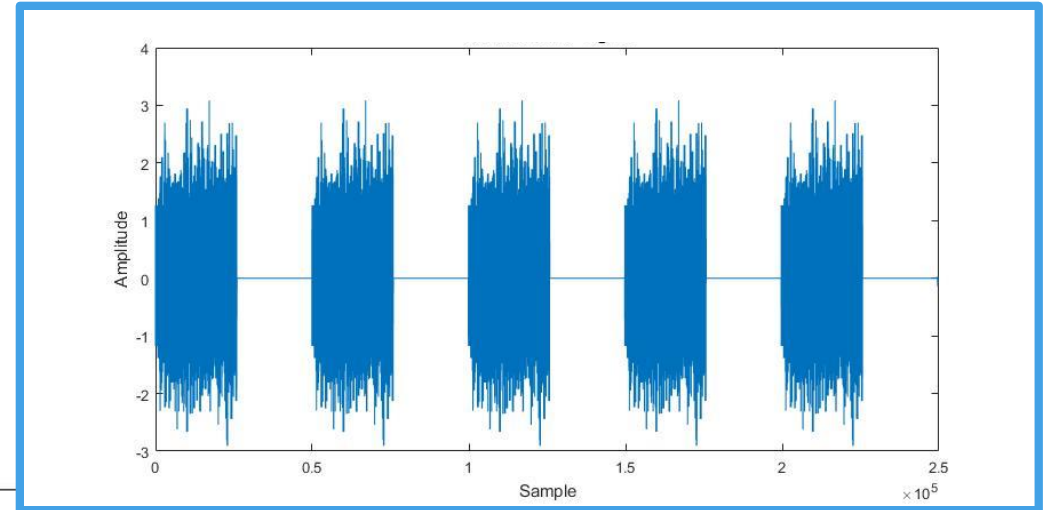
Implementation and Verification

- Generate portable, production-ready HDL and C
- Easily allow re-targeting to different hardware
- Flow to production testing and verification

Use Case

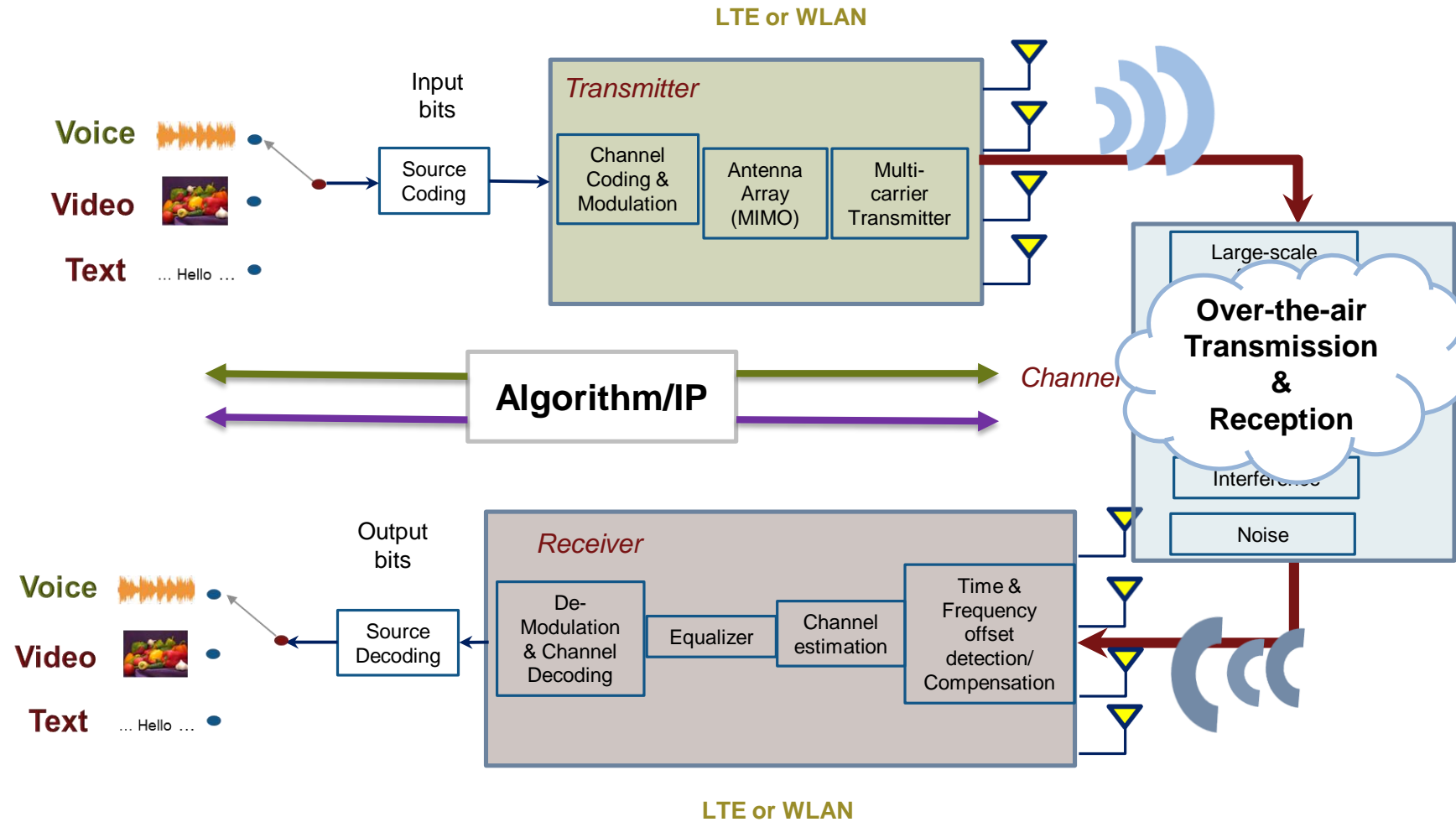
Waveform Generation

Testing & Verification

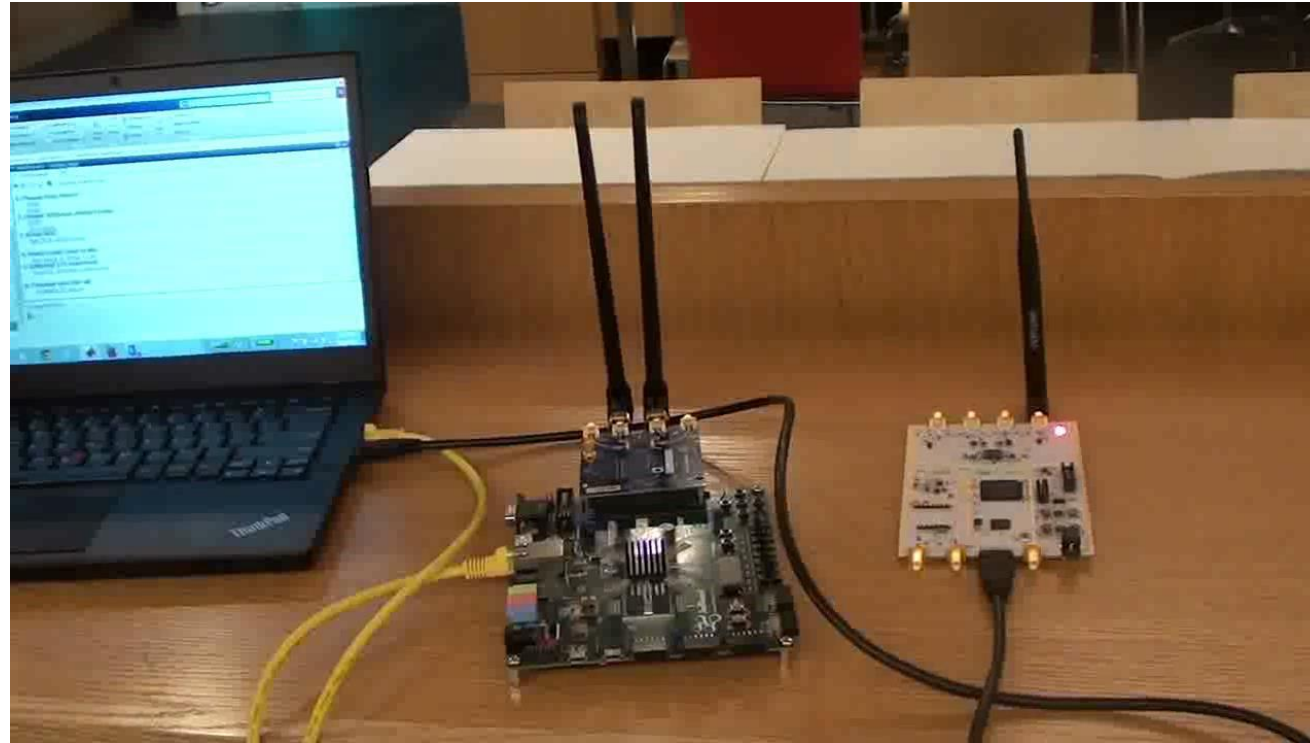


End-to-end simulation

Over-the-air testing



Physical connectivity to radio hardware



Software setup: Hardware support packages

The screenshot shows the MATLAB interface with the Support Package Installer window open. The window displays a list of support packages for the Communications System Toolbox. A red circle highlights the 'Xilinx Zynq-Based Radio' package in the list. A red arrow points from the 'Get Hardware Support Packages' option in the Add-Ons menu to the 'Xilinx Zynq-Based Radio' package in the installer window.

Support Package Installer

Select support package to install

Show: All (71)

Support for:

- Teledyne DALSA Spera Hardware
- Texas Instruments C2000
- Texas Instruments C2000 Concerto
- Texas Instruments C6000
- Total Phase Hardware I2C/SPI Interface
- USB Webcams
- USRP Radio
- Vector CAN Devices
- Wind River VxWorks
- Xilinx FPGA Boards
- Xilinx FPGA-Based Radio
- Xilinx Zynq-7000
- Xilinx Zynq-Based Radio

Support packages:

Action	Installed Version	Latest Version	Description	Required Base Product	Supported Host Platforms
1 <input checked="" type="checkbox"/> Install		15.1.0	Design and prototype SDR systems using Xilinx Zynq-based radio	Communications System Toolbox	Win32, Win64, ...

Installation folder: C:\MATLAB\SupportPackages\R2015a

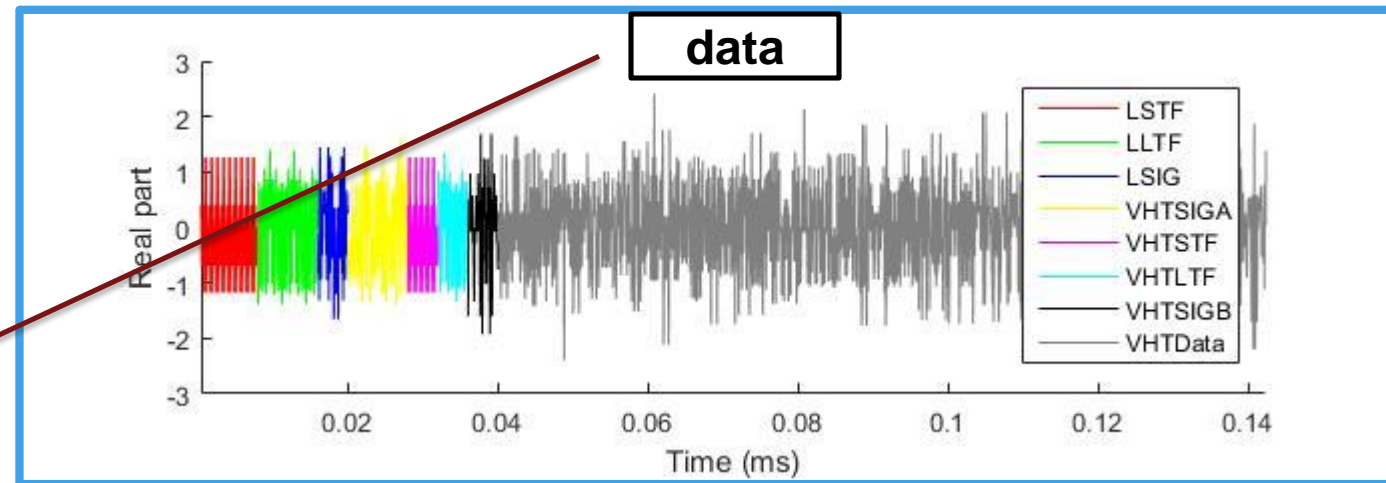
Buttons: < Back, Next >, Cancel, Help

Add-Ons Menu:

- Get Add-Ons
- Manage Add-Ons
- Package Toolbox
- Package App
- Get Hardware Support Packages
- Check for Product Updates

Exchange data between host computer and radio hardware

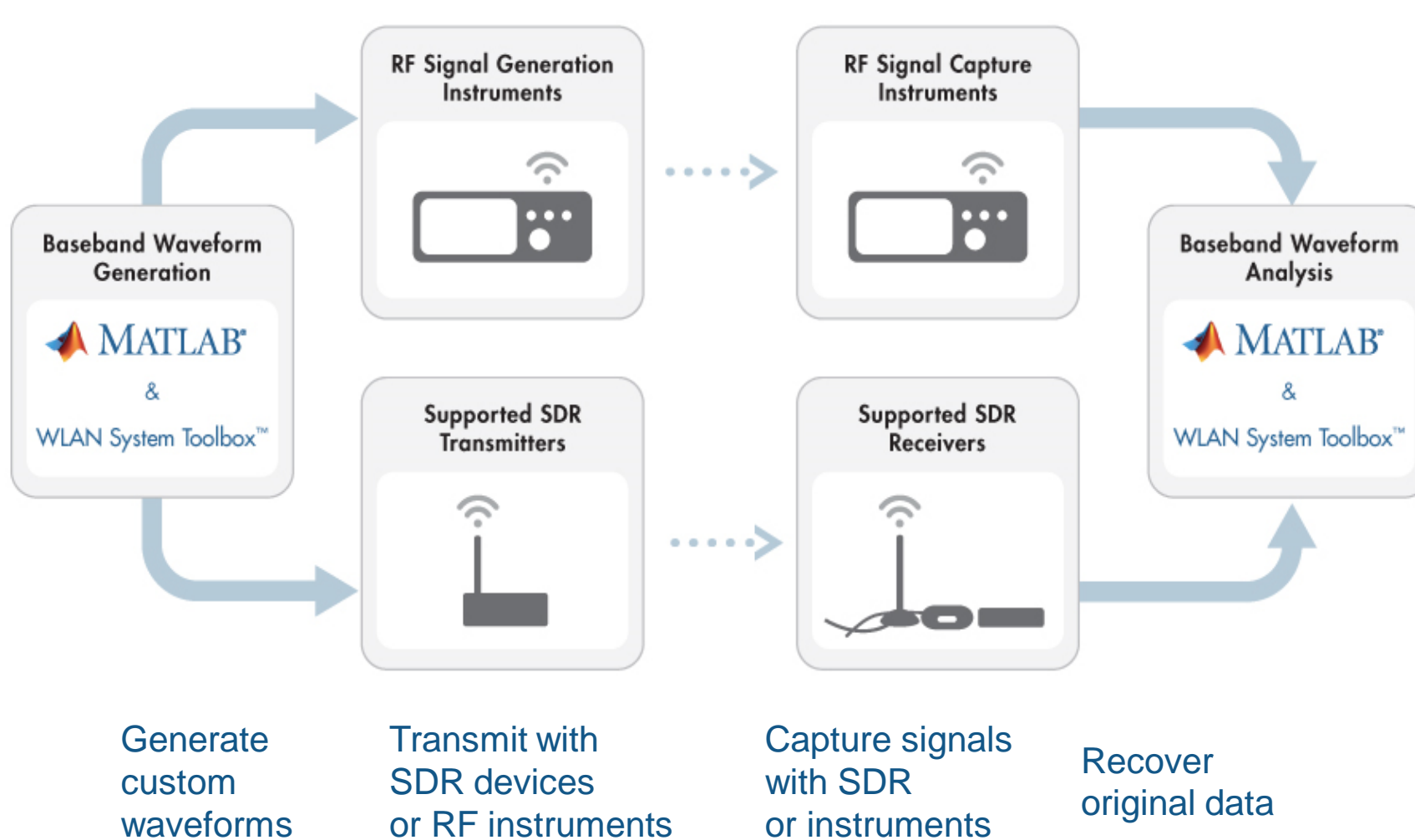
```
cfg = wlanVHTConfig;  
data = wlanWaveformGenerator(bits, cfg);
```



Command Window

```
>> usrpTx = comm.SDRuTransmitter;  
fx>> step(usrpTx, data);
```

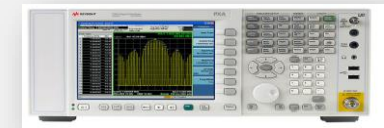
Hardware & Radio Connectivity



Range of supported hardware



RF Signal Generator



Spectrum Analyzer



Zynq Radio SDR



USRP SDR

Supported SDRs & RF instruments

RF Signal Generator



Zynq Radio SDR



USRP SDR



RF Spectrum Analyzer



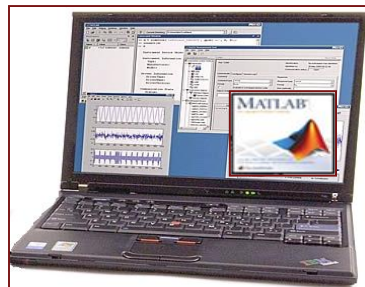
Zynq Radio SDR



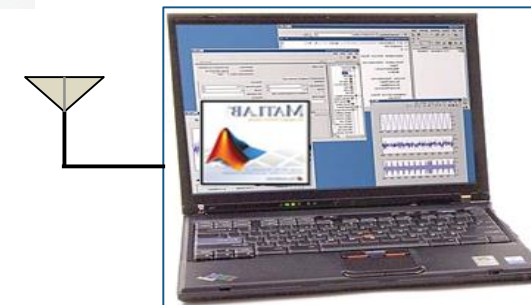
USRP SDR



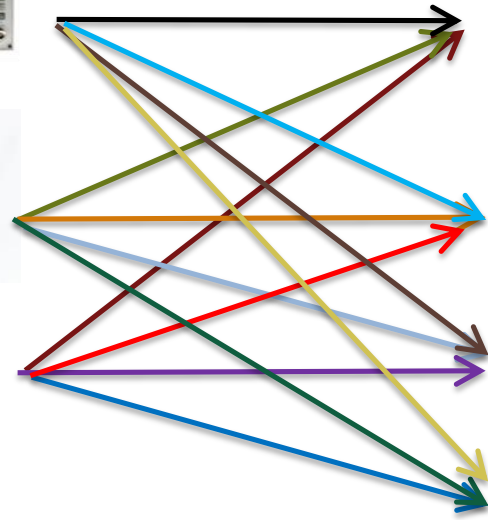
RTL SDR



Transmitter



Receiver



Use Case

H/W Prototyping Implementation

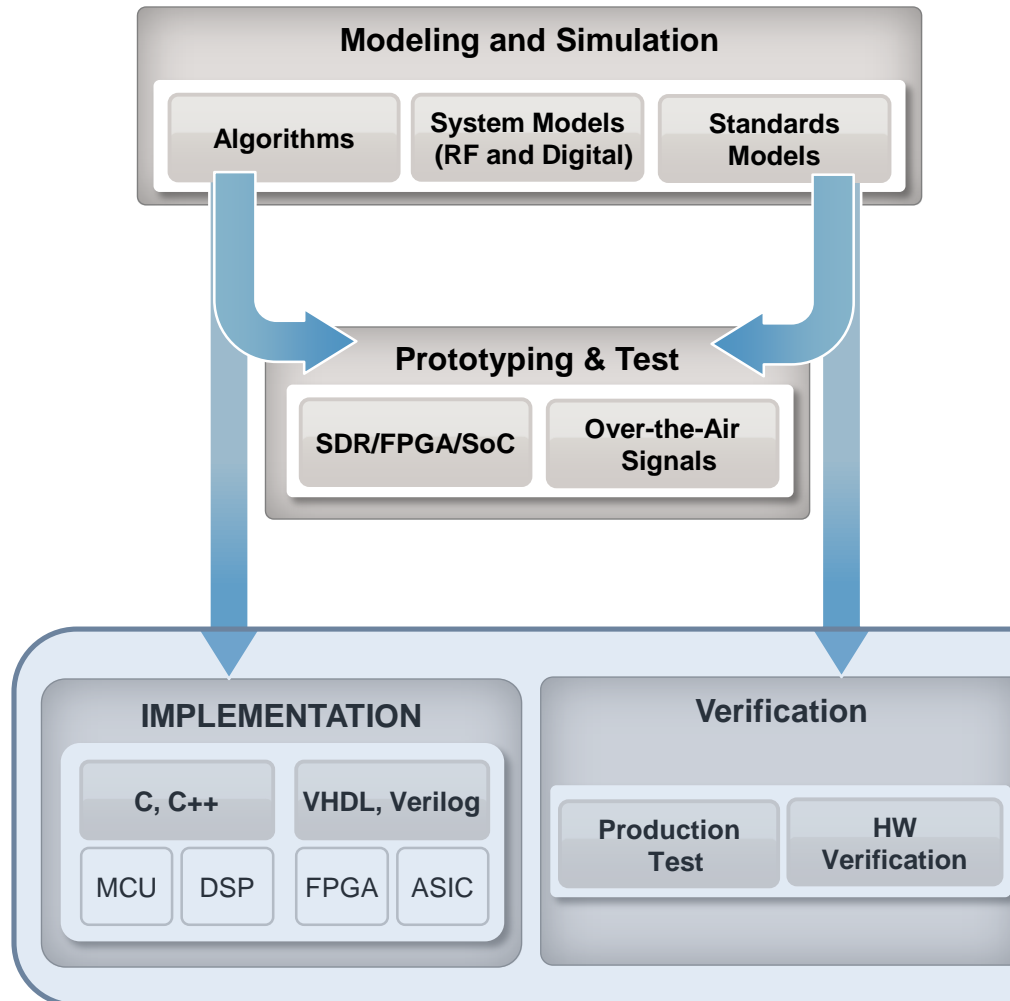
```

1  |-----
2  --
3  -- File Name: hdlsrc\hdlcoder_lteofdm_modDetect\LTE_Detector_HDL_pkg.vhd
4  -- Created: 2016-03-13 18:00:39
5  --
6  -- Generated by MATLAB 9.0 and HDL Coder 3.8
7  --
8  |-----
9
10
11  LIBRARY IEEE;
12  USE IEEE.std_logic_1164.ALL;
13  USE IEEE.numeric_std.ALL;
14
15  PACKAGE LTE_Detector_HDL_pkg IS
16      TYPE vector_of_signed16 IS ARRAY (NATURAL RANGE <>) OF signed(15 DOWNTO 0);
17      TYPE vector_of_signed18 IS ARRAY (NATURAL RANGE <>) OF signed(17 DOWNTO 0);
18      TYPE vector_of_signed20 IS ARRAY (NATURAL RANGE <>) OF signed(19 DOWNTO 0);
19      TYPE vector_of_signed22 IS ARRAY (NATURAL RANGE <>) OF signed(21 DOWNTO 0);
20      TYPE vector_of_unsigned8 IS ARRAY (NATURAL RANGE <>) OF unsigned(7 DOWNTO 0);
21      TYPE vector_of_unsigned4 IS ARRAY (NATURAL RANGE <>) OF unsigned(3 DOWNTO 0);
22      TYPE vector_of_unsigned6 IS ARRAY (NATURAL RANGE <>) OF unsigned(5 DOWNTO 0);
23      TYPE vector_of_signed24 IS ARRAY (NATURAL RANGE <>) OF signed(23 DOWNTO 0);
24      TYPE vector_of_unsigned3 IS ARRAY (NATURAL RANGE <>) OF unsigned(2 DOWNTO 0);
25      TYPE vector_of_signed25 IS ARRAY (NATURAL RANGE <>) OF signed(24 DOWNTO 0);
26      TYPE vector_of_signed32 IS ARRAY (NATURAL RANGE <>) OF signed(31 DOWNTO 0);
27      TYPE vector_of_std_logic_vector32 IS ARRAY (NATURAL RANGE <>) OF std_logic_vector(31 DOWNTO 0);

```



Model-Based Design: As Algorithms move toward Implementations



Modeling and Simulation

- MATLAB has a widespread use in algorithm design
- Used also for multi-domain simulation: Digital + RF
- Increasingly provides wireless standard models

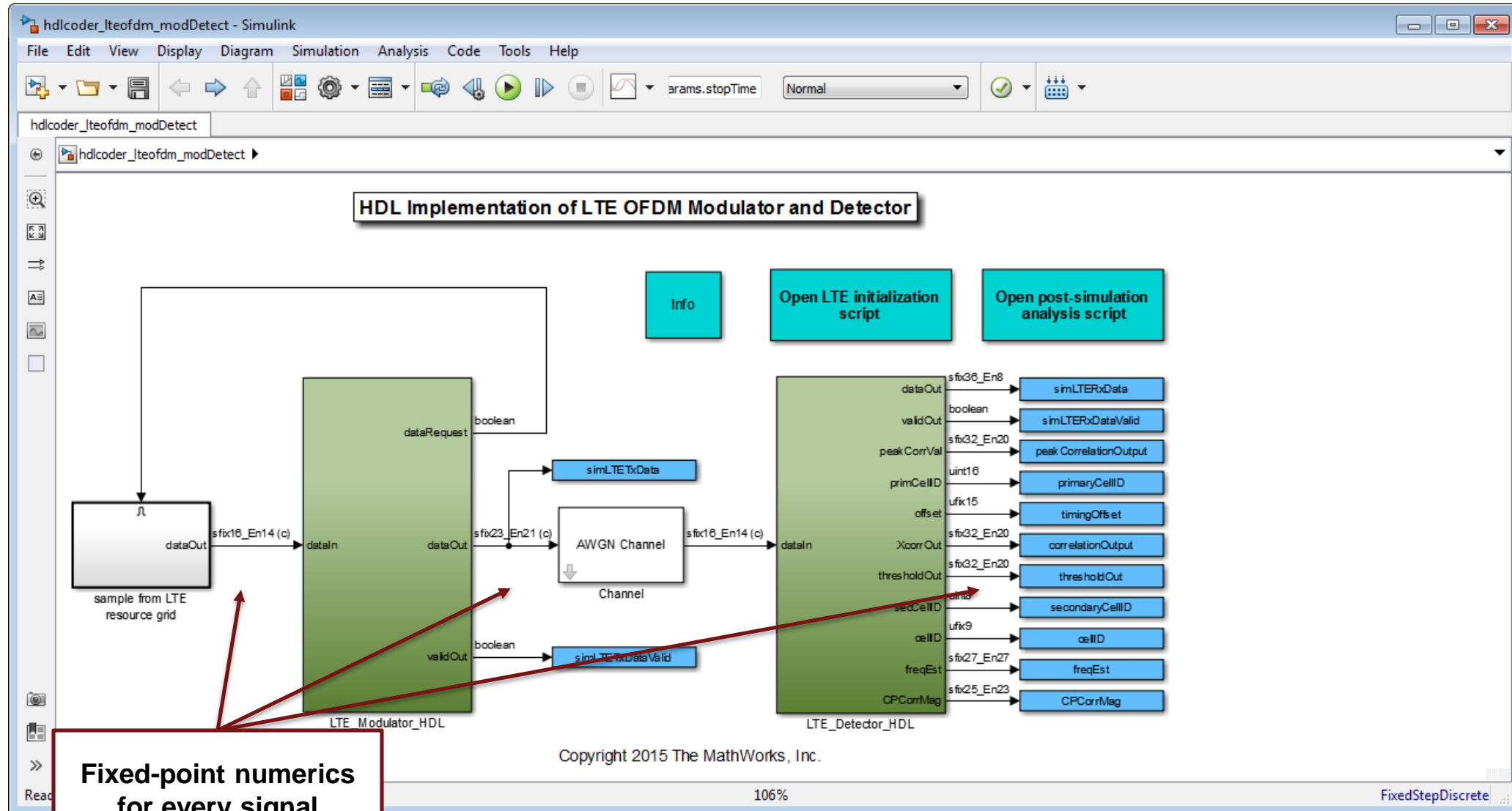
Prototyping and Testing

- Test with a range of SDR and instrument hardware
- Rapid prototyping with code generation

Implementation and Verification

- Generate portable, production-ready HDL and C
- Easily allow re-targeting to different hardware
- Flow to production testing and verification

Implementation-ready Algorithms: fixed-point



Fixed-point algorithm design

1. Set-up simulation flow
2. Express your floating-point algorithm
 - focus on algorithmic integrity, proof of concept
3. Simulate (floating-point)
 - iterate on algorithm trade-offs, validate against requirements
4. Convert design to fixed-point
 - Focus of design viability based on implementation constraints
5. Simulate (fixed-point)
 - iterate on implementation trade-offs, validate against original requirements
6. Generate code for implementation
7. Validate and verify design after deployment

C/C++

RTL

Use automatic fixed-point conversion tools

The image displays the Simulink environment with a model titled `hdlcoder_lteofdm_modDetect`. The model diagram shows a block labeled `sample from LTE resource grid` connected to `dataOut`, which then feeds into `dataIn` of the `LTE_Modulator_HDL` block. The `LTE_Modulator_HDL` block outputs `dataRequest` (boolean) and `dataOut`, which is then processed by `LTE_Detector_HDL`. The `LTE_Detector_HDL` block outputs `dataOut` and `sample from LTE resource grid`.

The **Fixed-Point Tool** window is open, showing the **Model Hierarchy** on the left. The hierarchy includes `Fixed-Point Tool Root`, `hdlcoder_lteofdm_modDetect`, `AnalysisScriptSubsys`, `Channel`, `InitScriptSubsys`, `LTE_Detector_HDL`, `LTE_Modulator_HDL`, and `sample from LTE resource grid`.

The **Contents of: Compare To Zero 1** table is visible, showing columns for `Name`, `Run`, `CompiledDT`, `SpecifiedDT`, `SimMin`, and `SimMax`.

The **Fixed-point preparation** section includes the **Fixed-Point Advisor** and **Configure model settings** options:

- ☐ Range collection using double override
- ☐ Range collection with specified data types
- ☐ Remove overrides and disable range collection

The **Advanced settings** section includes the **Range collection** settings:

- Run name: `Run 1`
- ☐ Simulate model
- ☐ Merge results from multiple simulations
- Derive ranges for: `System Under Design`

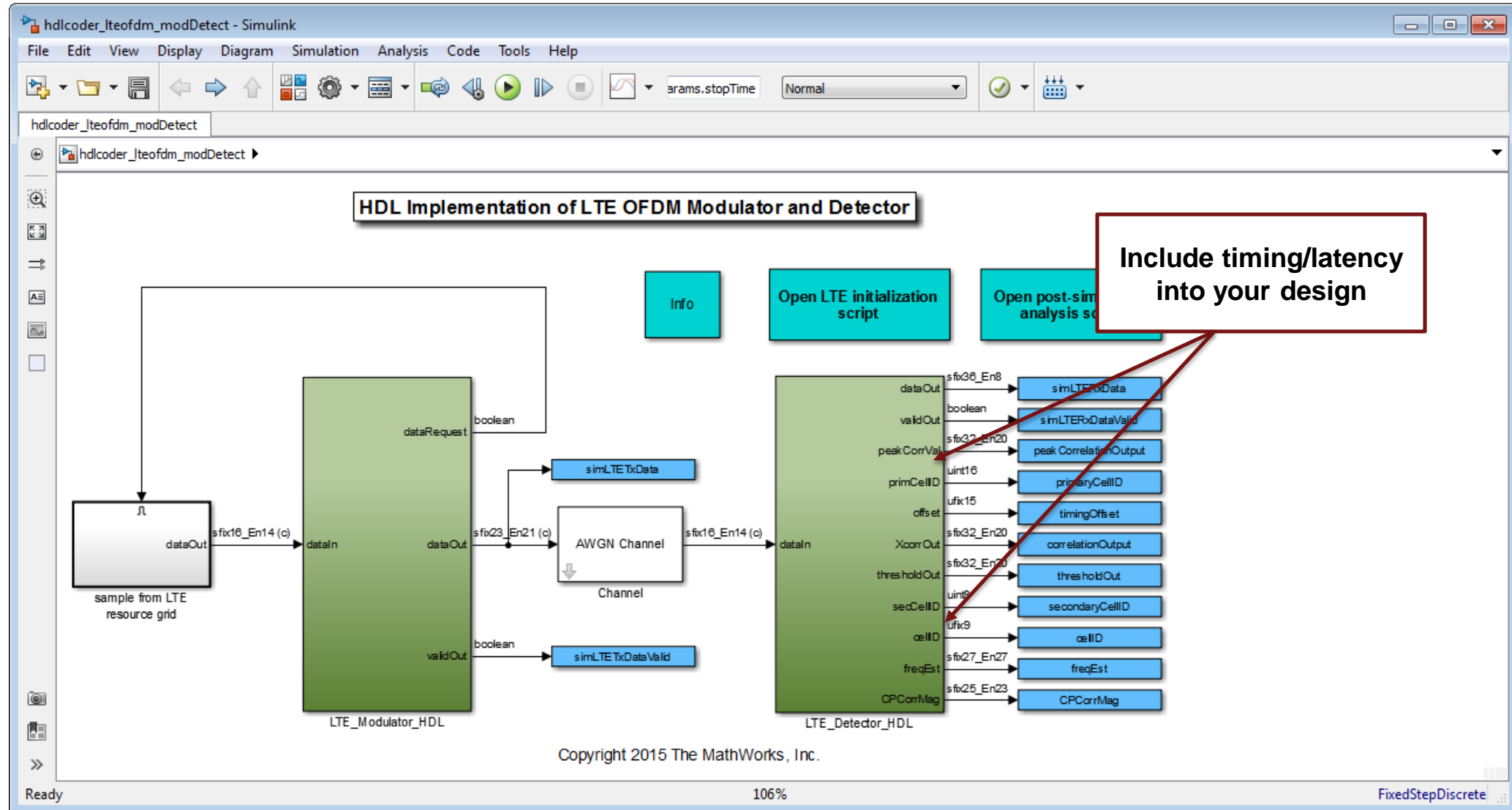
The **Automatic data typing** section includes the **Propose** settings:

- ☒ Signedness
- ☐ Word length
- ☒ Fraction length

Red arrows point from the **Automatic/supervised fixed-point conversion tools** text box to the **Fixed-Point Tool** window, specifically to the **Configure model settings** and **Advanced settings** sections.

Copyright 2015 The MathWorks, Inc.

Implementation-ready Algorithms: timing & parallelism



Target your SDR for HDL Code Generation

The image displays the Simulink environment with a model titled "hdlcoder_iteofdm_modDetect". The model diagram shows the HDL Implementation of LTE OFDM Modulator and Detector. Key components include:

- sample from LTE resource grid**: A block that outputs `dataOut` (sfix16_En14 (c)) to `dataIn`.
- LTE_Modulator_HDL**: A large green block that receives `dataIn` and outputs `dataRequest` (boolean), `dataOut` (sfix23_En21 (c)), and `validOut` (boolean).
- simLTE_TxData**: A block that receives `dataRequest` and outputs `dataOut` (sfix23_En21 (c)) to the **AWGN Channel**.
- AWGN Channel**: A block that receives `dataOut` and outputs `dataIn` (sfix16_En14 (c)) to **LTE_Detector_HDL**.
- simLTE_TxDataValid**: A block that receives `validOut` and outputs `dataIn` (sfix16_En14 (c)) to **LTE_Detector_HDL**.

The HDL Workflow Advisor window is open, showing the task list:

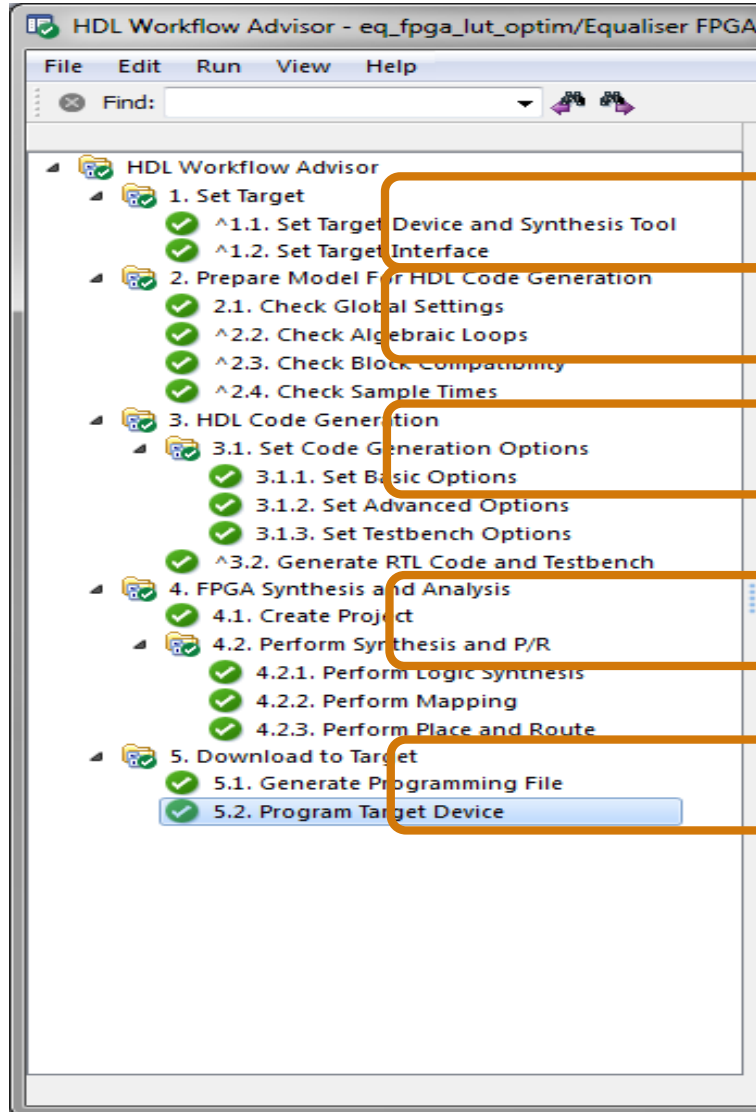
- 1. Set Target
 - 1.1. Set Target Device and Synthesis Tool
- 2. Prepare Model For HDL Code Generation
 - 2.1. Check Global Settings
 - 2.2. Check Algebraic Loops
 - 2.3. Check Block Compatibility
 - 2.4. Check Sample Times
 - 2.5. Check FPGA-in-the-Loop
- 3. HDL Code Generation
 - 3.1. Set Code Generation Options
 - 3.2. Generate RTL Code and Synthesize
- 4. FPGA-in-the-Loop Implementation

The **1.1. Set Target Device and Synthesis Tool** task is selected, showing the following configuration:

- Analysis** (^Triggers Update Diagram)
- Set Target Device and Synthesis Tool for HDL code generation**
- Input Parameters**
 - Target workflow: **FPGA-in-the-Loop**
 - Target platform: **Choose a platform** (with **Launch Board Manager** button)
 - Synthesis tool: **No synthesis tool available on system path** (with **Refresh** button)
 - Family: **[Empty]**
 - Device: **[Empty]**
 - Package: **[Empty]**
 - Speed: **[Empty]**
 - Project folder: **hdl_prj** (with **Browse...** button)
- Run This Task** button
- Result**: **Not Run**
- Click Run This Task.**

Copyright 2015 The MathWorks, Inc.

HDL Workflow Advisor



Select ASIC, FPGA, Or FPGA Board Target

Prepare Model For HDL Code Generation

Generate HDL Code

Physical Design and Critical Path Highlighting

Program FPGA

SDR Hardware Verification and Prototyping

- Requirements
 - Hardware testbed to verify designs with live radio signals
 - Support for a slew of waveforms
 - Support for standards such as LTE & Wi-Fi
- Quick prototyping on SDR
 - Incremental implementation of algorithms operating on baseband signal
 - Speed/size/memory assessment & optimization
 - Ultimately porting entire transmitter or receiver algorithms on FPGA



Paving the way for the future

- Same algorithm no longer manually implemented for each target platform
- Re-targeting
 - Represent it such that it is ported automatically on various SDRs
- System-level modeling
 - Integrate all low-level implementation details (fixed-point, latency, ...) into System model
 - Elaborate system model and verify in software before targeting the hardware

Main goal

- Automatic porting/prototyping of algorithms to multicore devices of heterogeneous nature