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

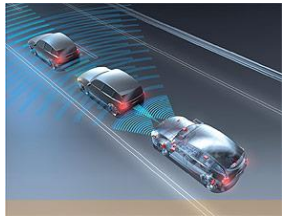
- rapid prototyping of wireless algorithms
- Evaluate radio hardware in real-world conditions











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

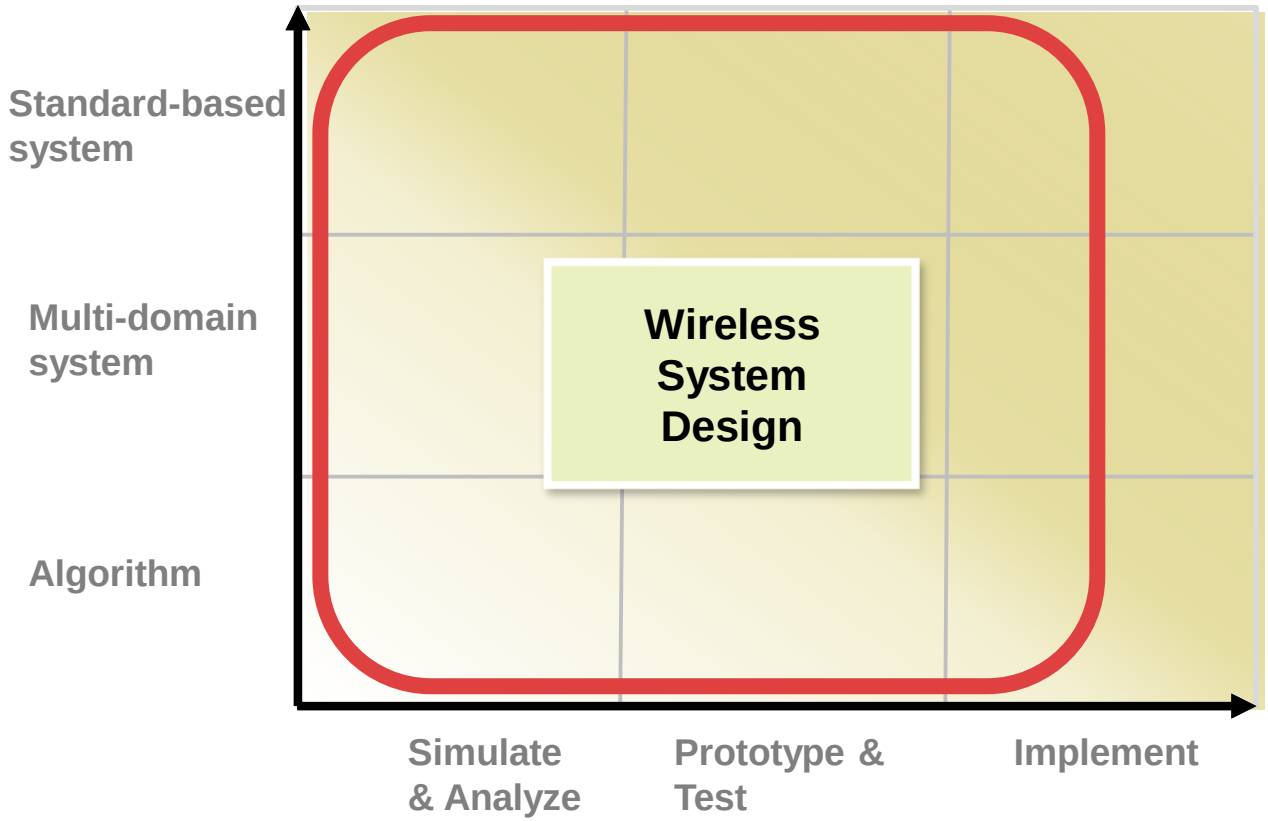
Wireless Prototyping

Who are the users?

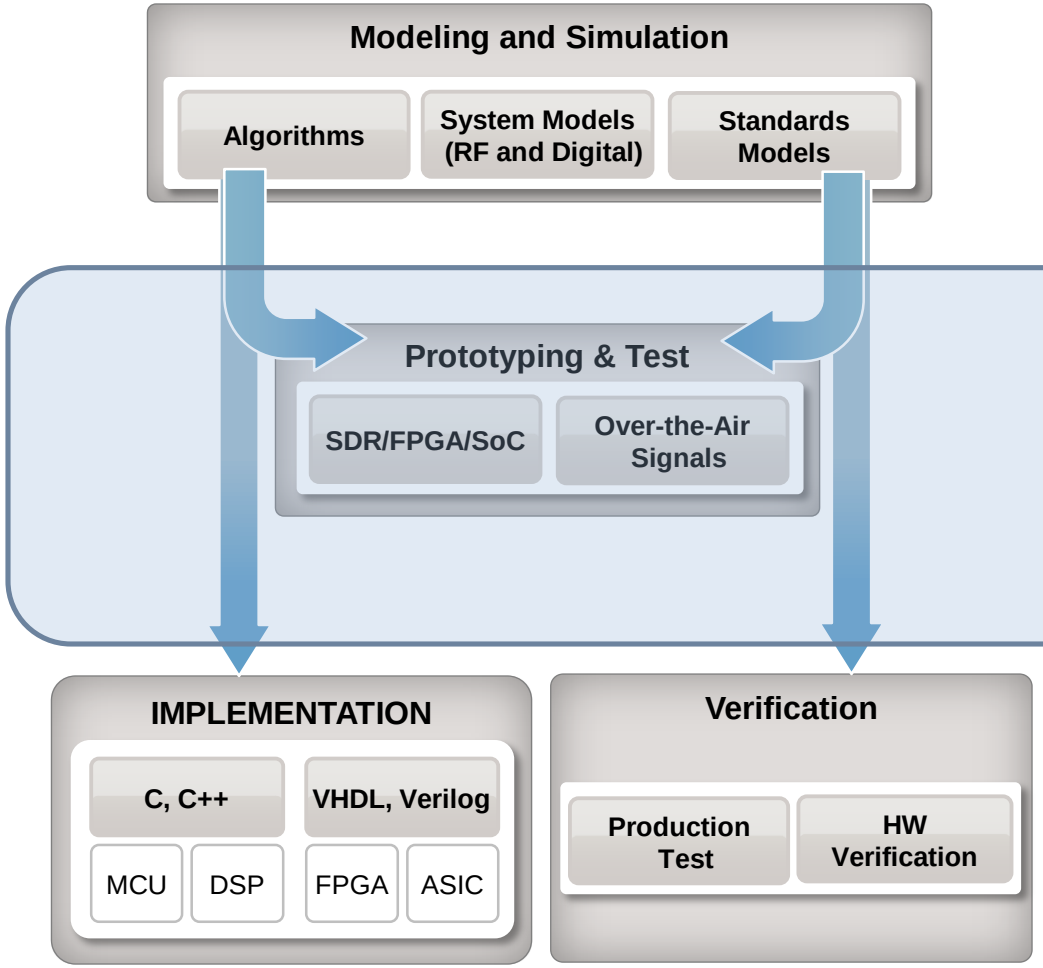
- Communications engineer
- RF System engineer
- Test or validation engineer

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

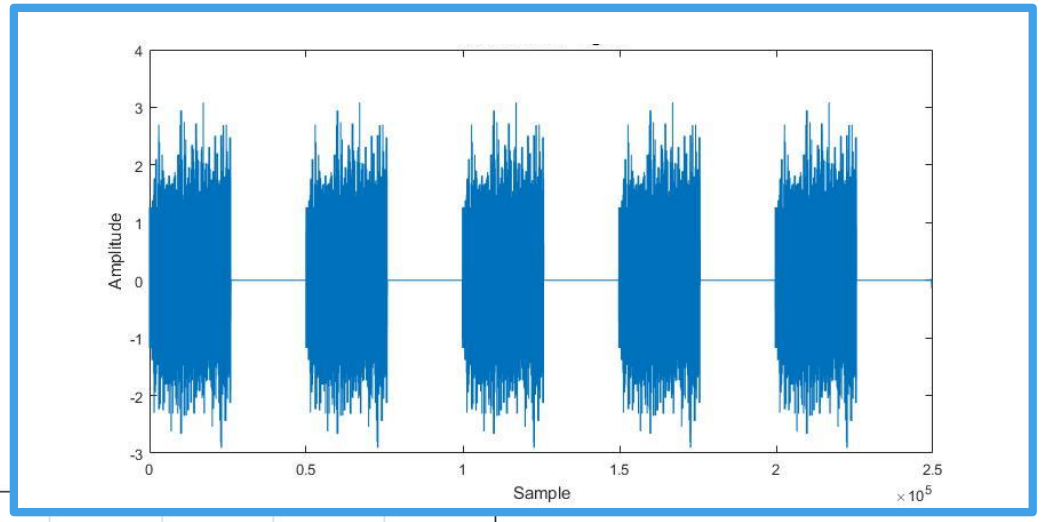
Implementation and Verification

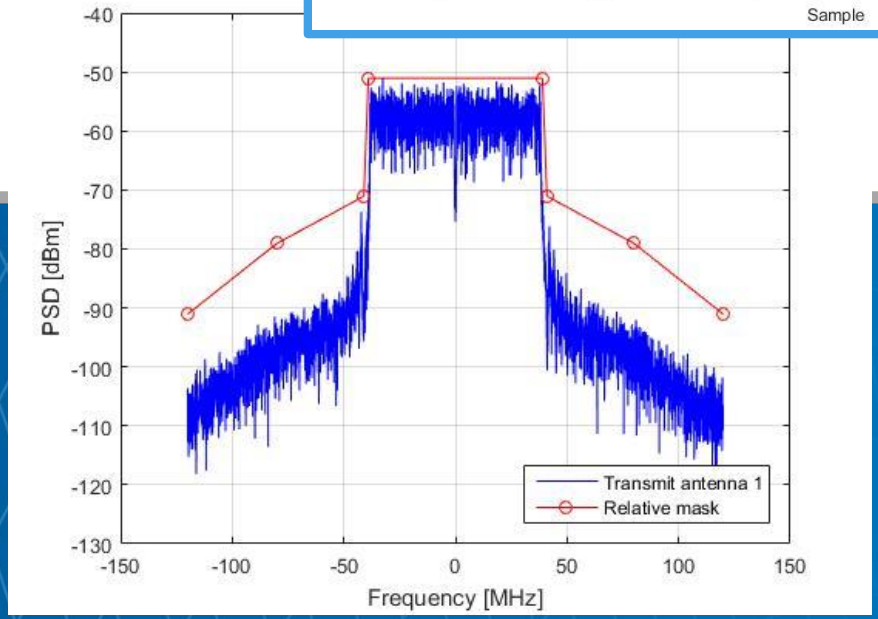
- Generate portable, production-ready HDL and C
- Easily allow re-targeting to different hardware
- Flow to production testing and verification

Use Case

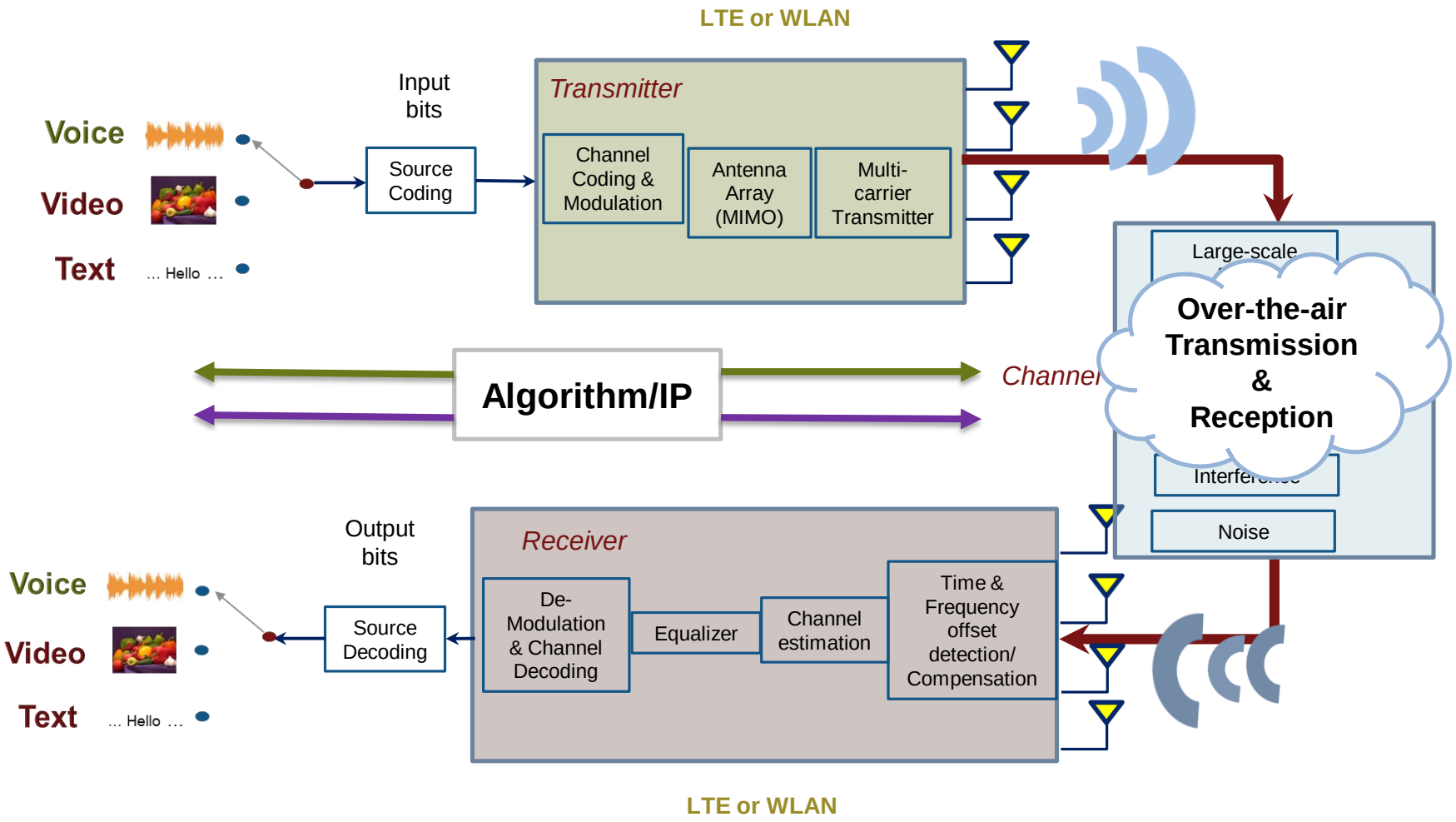
Waveform Generation

Testing & Verification

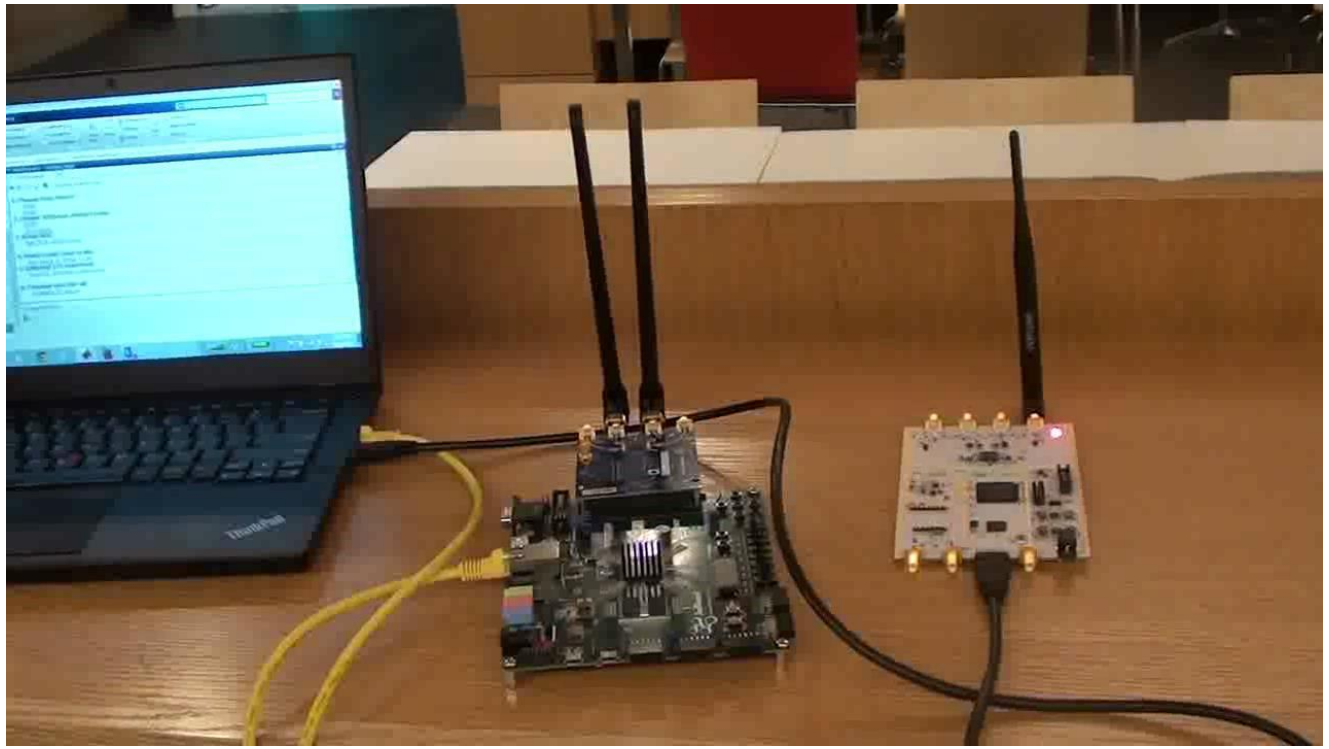




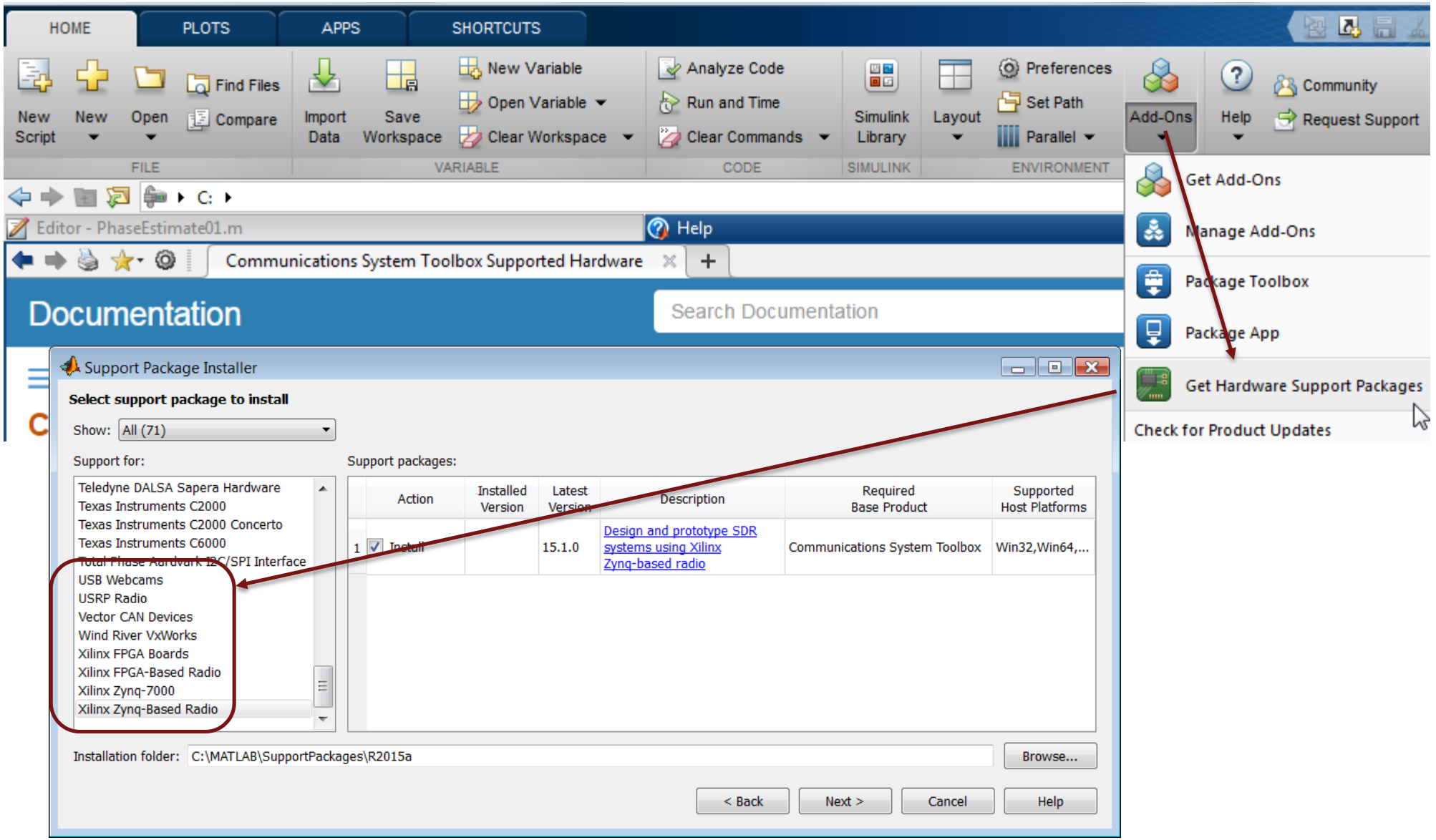
End-to-end simulation
Over-the-air testing



Physical connectivity to radio hardware

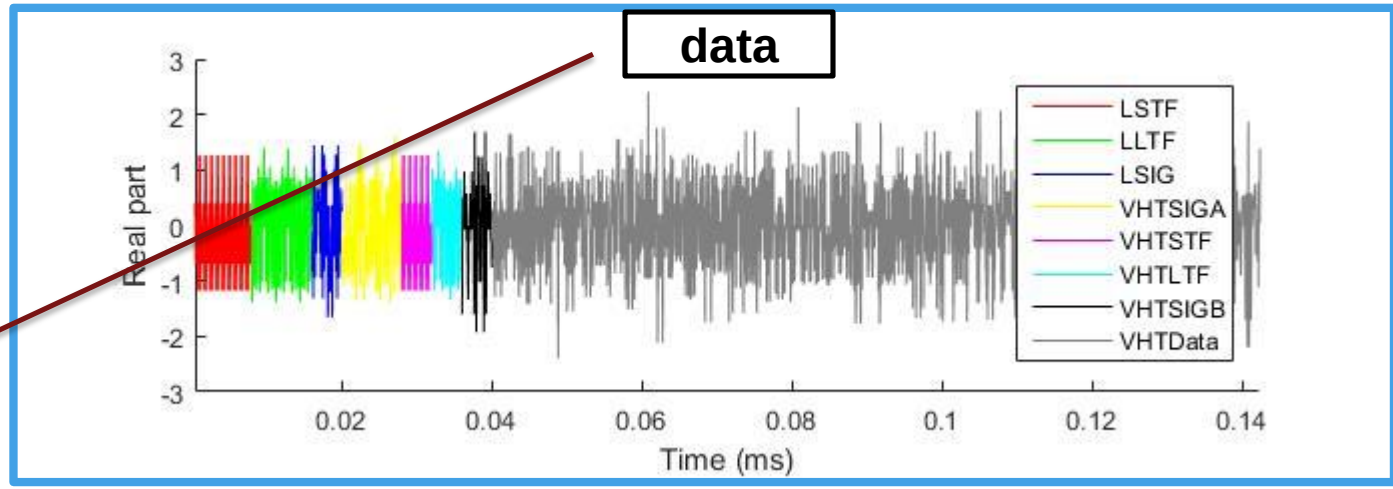


Software setup: Hardware support packages



Exchange data between host computer and radio hardware

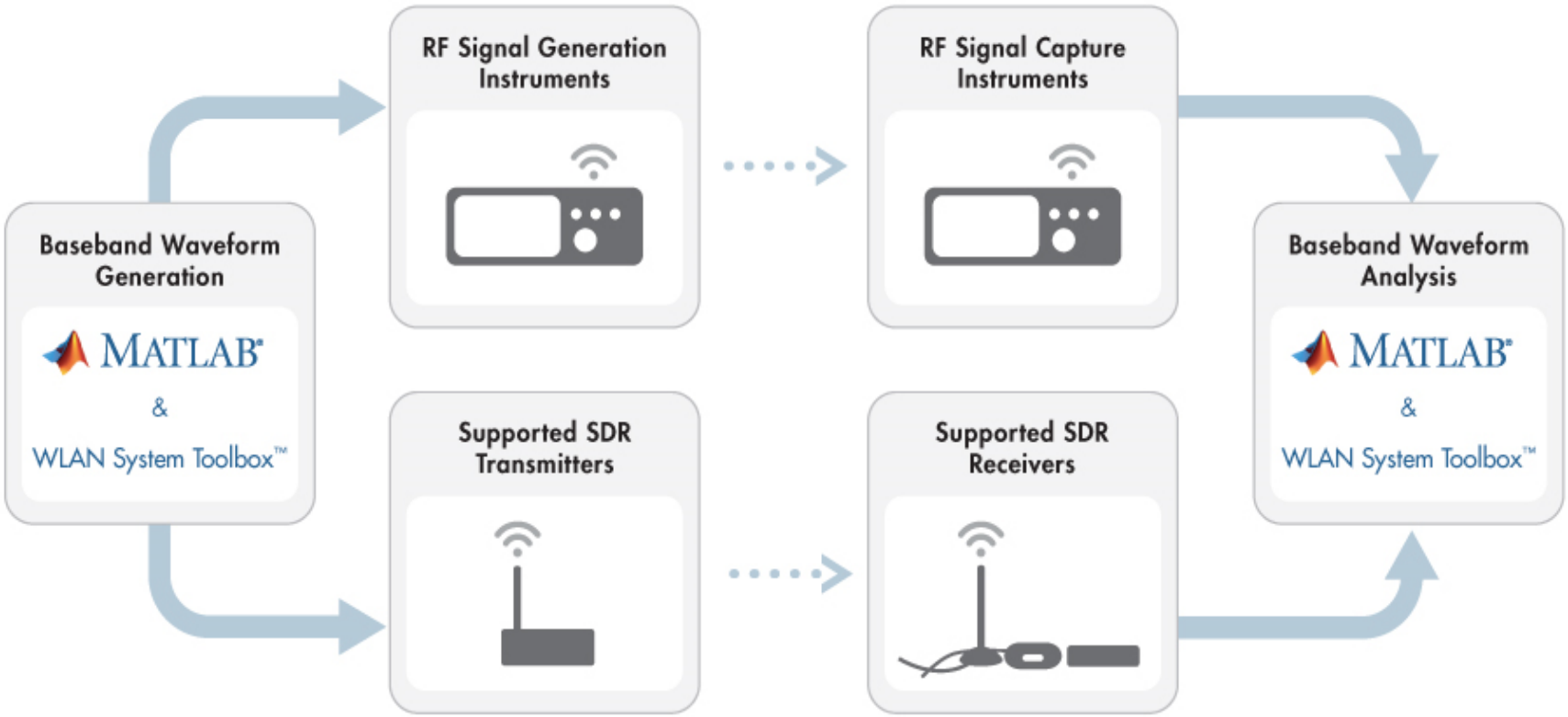
```
cfg = wlanVHTConfig;
data = wlanWaveformGenerator(bits, cfg);
```



Command Window

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

Hardware & Radio Connectivity



Generate custom waveforms

Transmit with SDR devices or RF instruments

Capture signals with SDR or instruments

Recover original data

Range of supported hardware

RF Signal Generator

Spectrum Analyzer

Zynq Radio SDR

USRP SDR

Supported SDRs & RF instruments

RF Signal Generator

Zynq Radio SDR

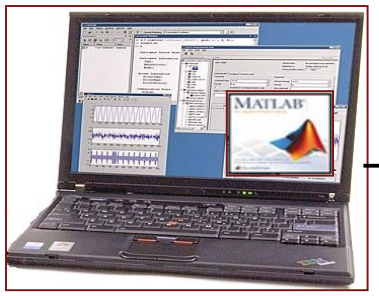
USRP SDR

RF Spectrum Analyzer

Zynq Radio SDR

USRP SDR

RTL SDR



Transmitter

Receiver

Use Case

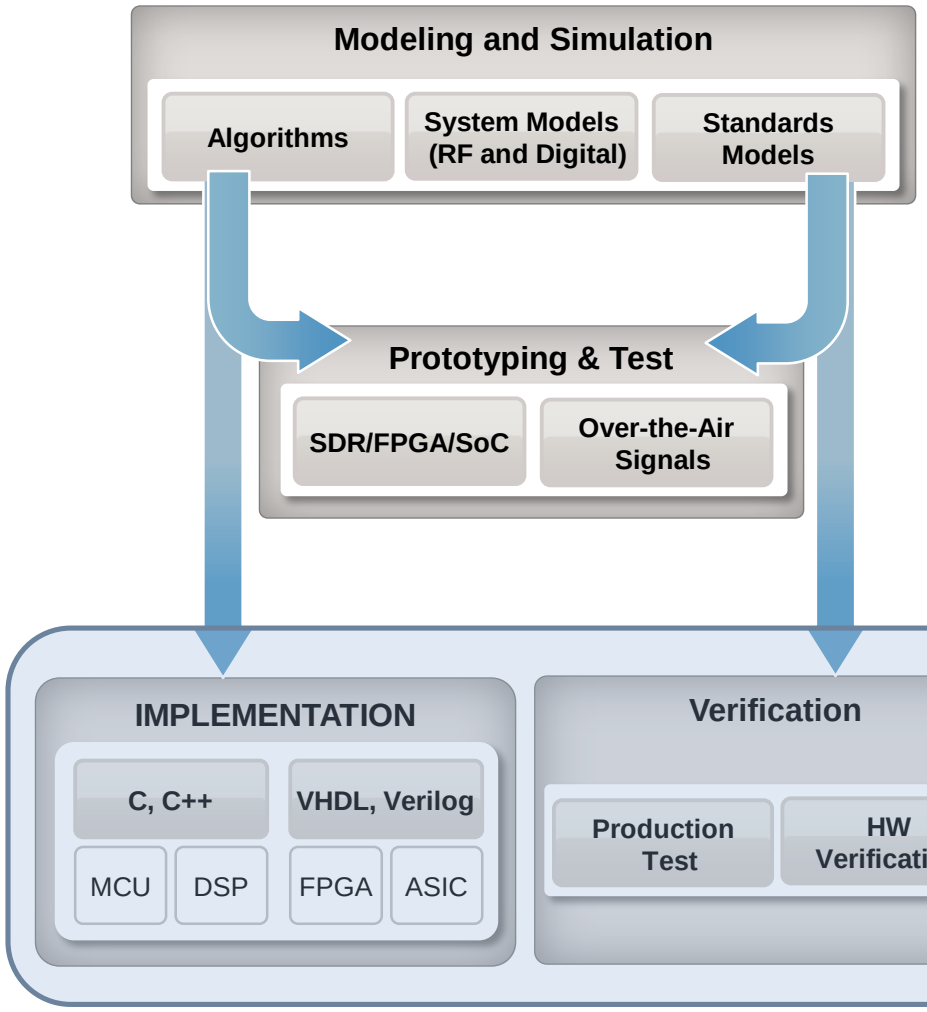
H/W Prototyping Implementation

```
-- -------------------------------------------------------------
--
-- File Name: hdlsrc\hdlcoder_lteofdm_modDetect\LTE_Detector_HDL_pkg.vhd
-- Created: 2016-03-13 18:00:39
--
-- Generated by MATLAB 9.0 and HDL Coder 3.8
--
-- -------------------------------------------------------------


LIBRARY IEEE;
USE IEEE.std_logic_1164.ALL;
USE IEEE.numeric_std.ALL;

PACKAGE LTE_Detector_HDL_pkg IS
  TYPE vector_of_signed16 IS ARRAY (NATURAL RANGE <>) OF signed(15 DOWNTO 0);
  TYPE vector_of_signed18 IS ARRAY (NATURAL RANGE <>) OF signed(17 DOWNTO 0);
  TYPE vector_of_signed20 IS ARRAY (NATURAL RANGE <>) OF signed(19 DOWNTO 0);
  TYPE vector_of_signed22 IS ARRAY (NATURAL RANGE <>) OF signed(21 DOWNTO 0);
  TYPE vector_of_unsigned8 IS ARRAY (NATURAL RANGE <>) OF unsigned(7 DOWNTO 0);
  TYPE vector_of_unsigned4 IS ARRAY (NATURAL RANGE <>) OF unsigned(3 DOWNTO 0);
  TYPE vector_of_unsigned6 IS ARRAY (NATURAL RANGE <>) OF unsigned(5 DOWNTO 0);
  TYPE vector_of_signed24 IS ARRAY (NATURAL RANGE <>) OF signed(23 DOWNTO 0);
  TYPE vector_of_unsigned3 IS ARRAY (NATURAL RANGE <>) OF unsigned(2 DOWNTO 0);
  TYPE vector_of_signed25 IS ARRAY (NATURAL RANGE <>) OF signed(24 DOWNTO 0);
  TYPE vector_of_signed32 IS ARRAY (NATURAL RANGE <>) OF signed(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

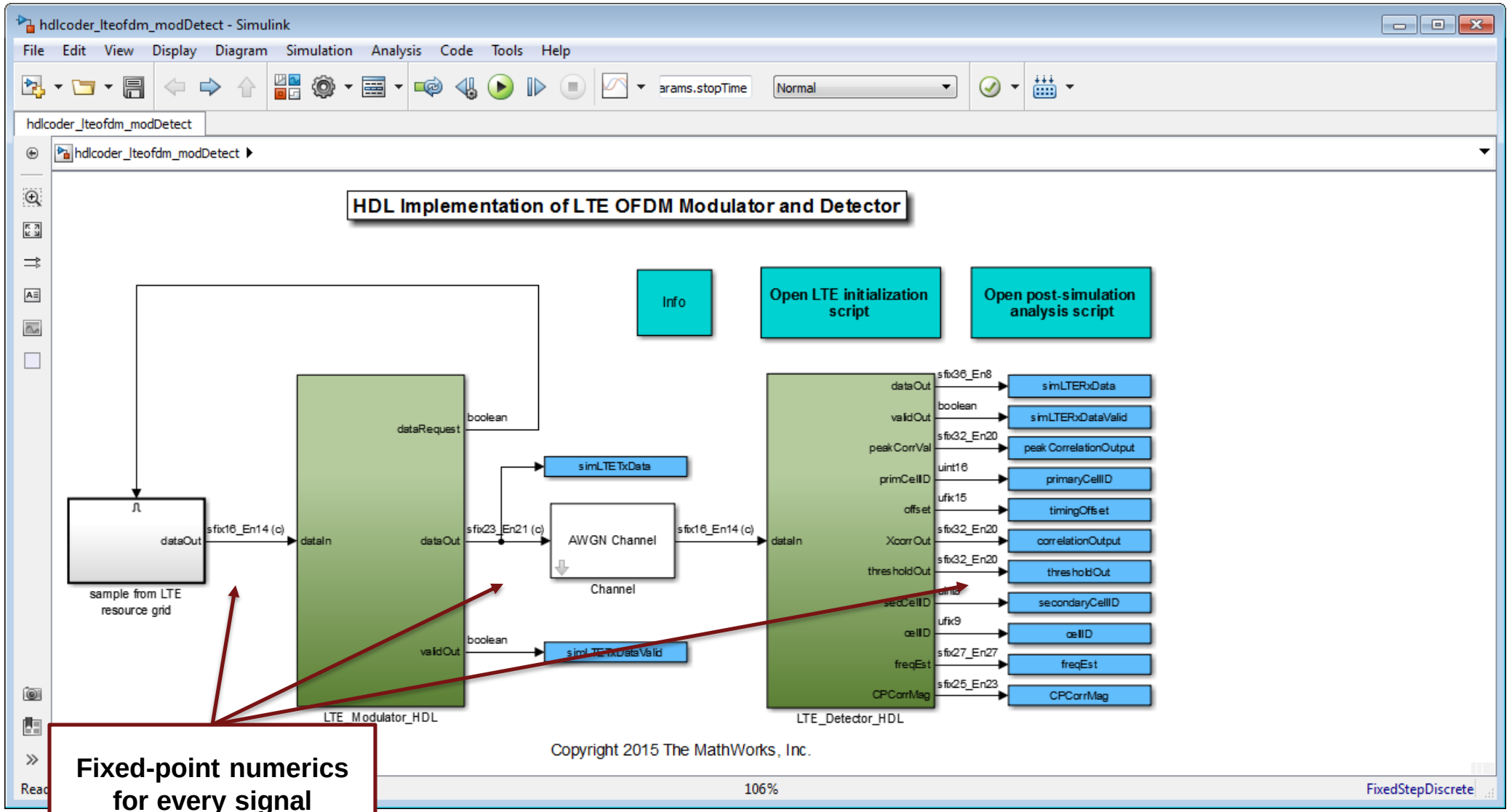
Prototyping and Testing

- 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



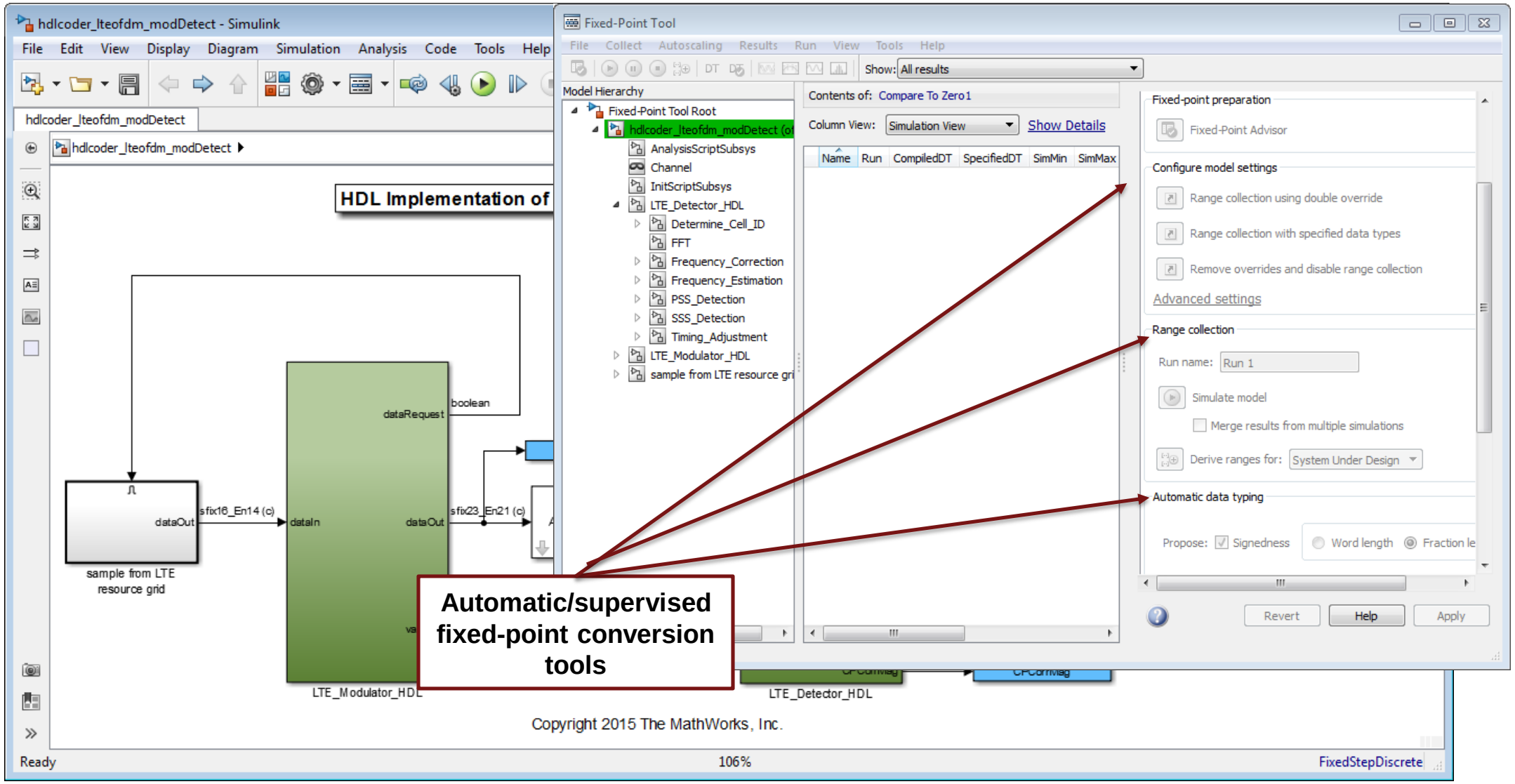
Fixed-point algorithm design

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

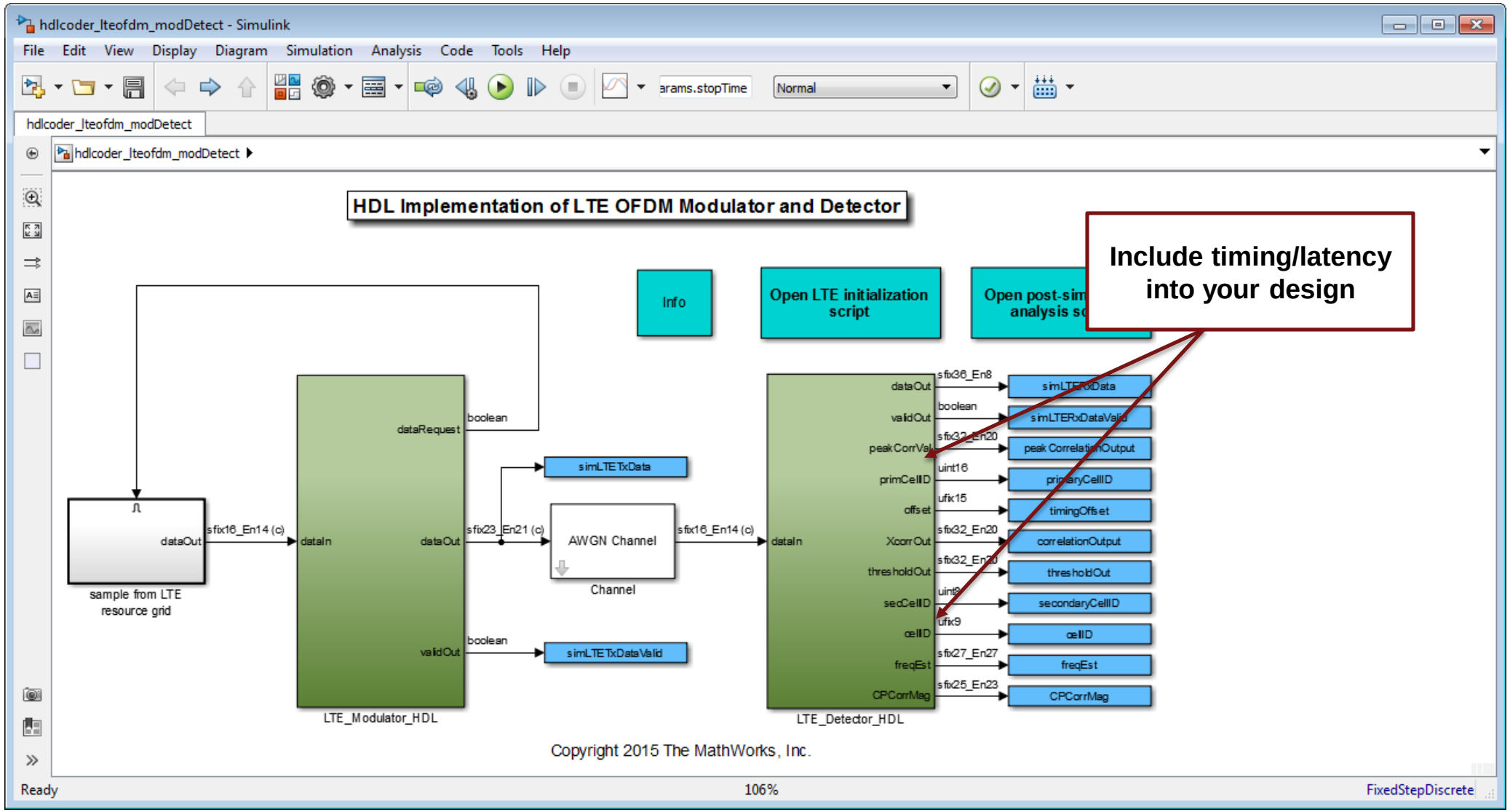
4. Convert design to fixed-point
   - Focus of design viability based on implementation constraints
5. Simulate (fixed-point)
   - iterate on implementation trade-offs, validate against original requirements

6. Generate code for implementation — ***C/C++***
7. Validate and verify design after deployment — ***RTL***

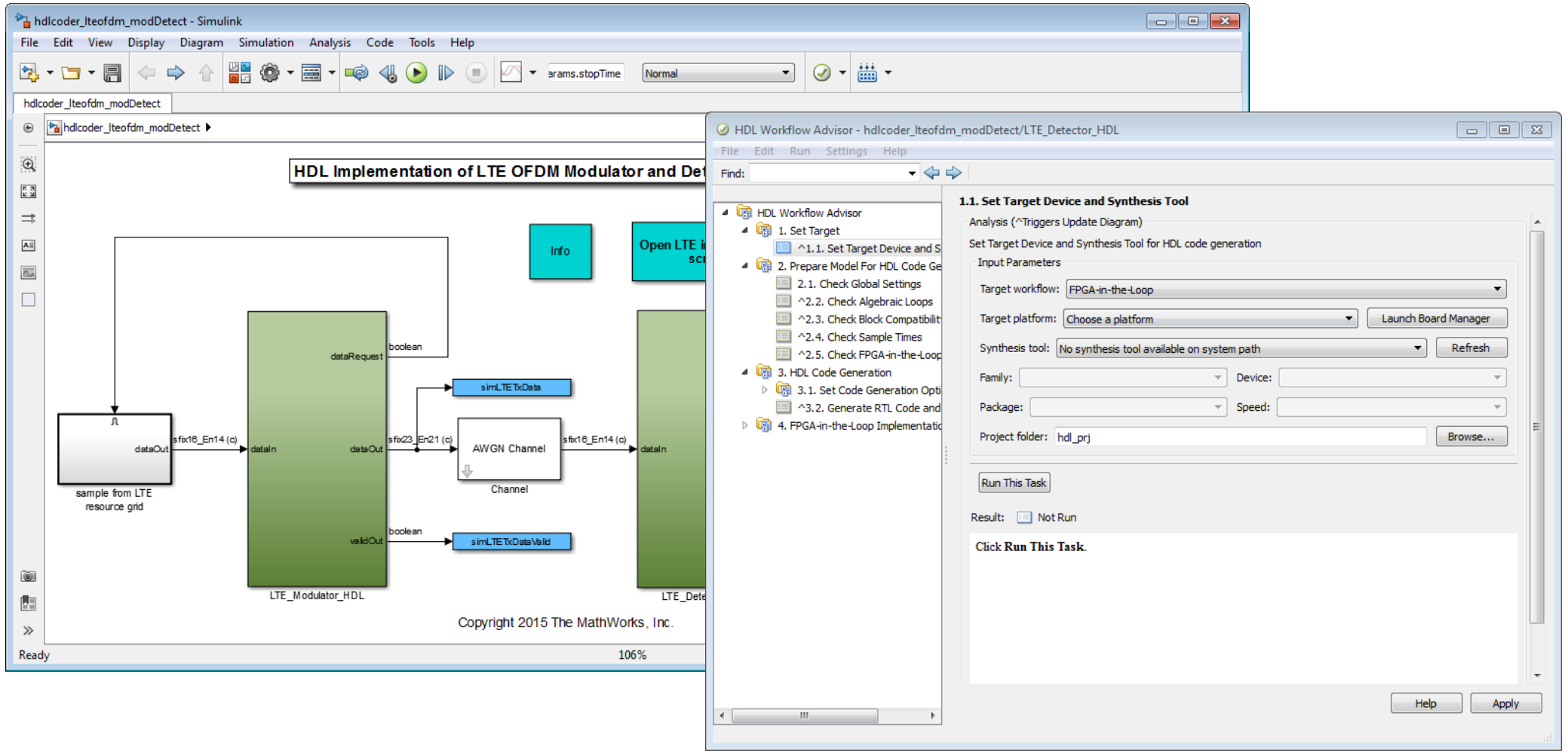
Use automatic fixed-point conversion tools



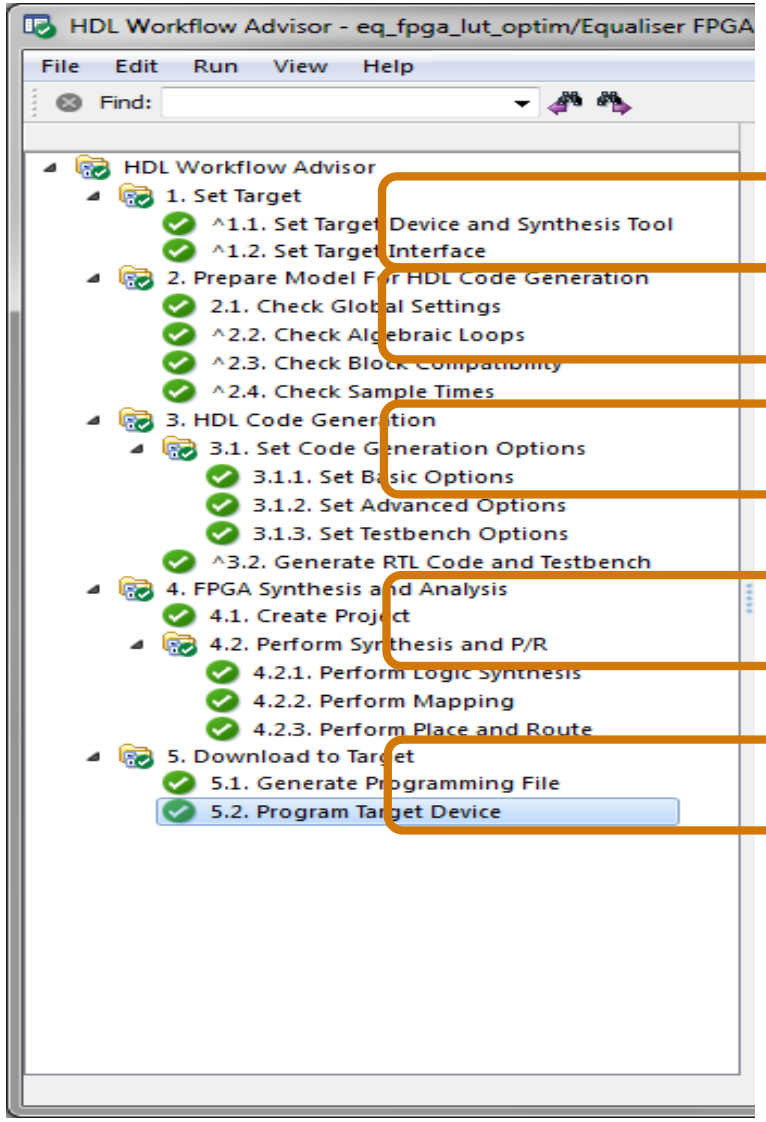
Implementation-ready Algorithms: timing & parallelism



Target your SDR for HDL Code Generation



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