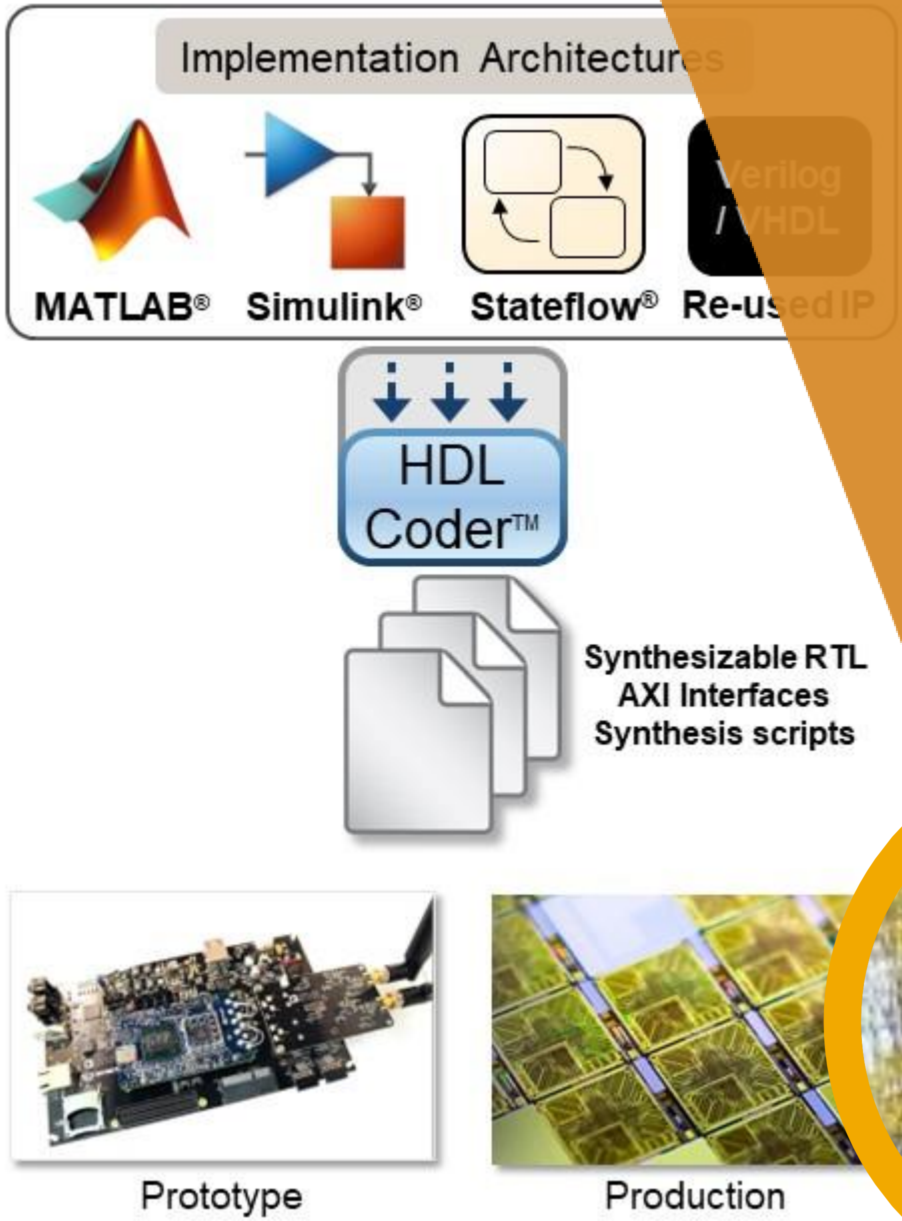


Implementation Architectures

MATLAB® Simulink® Stateflow® Re-used IP

Verilog / VHDL

HDL Coder™

Synthesizable RTL
AXI Interfaces
Synthesis scripts

Prototype

Production

MATLAB and Simulink

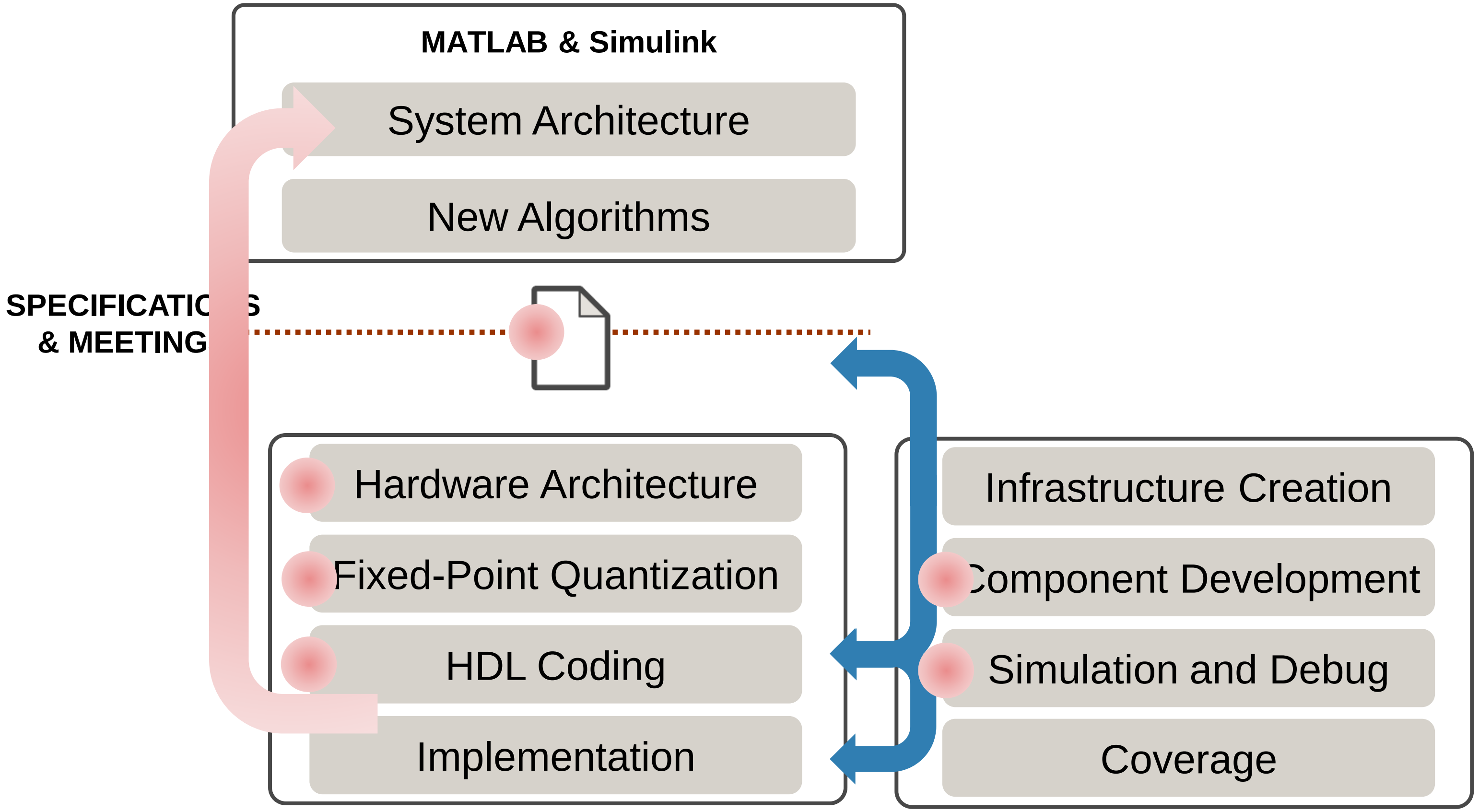
FPGA and SoC Development for Intel

Noam Levine

[title]

Disconnected Workflows Limit Innovation

- Miscommunication
- Limited Exploration
- Manual, error-prone
- Cross-team iteration



MATLAB & Simulink

System Architecture

New Algorithms

SPECIFICATIC S
& MEETING

Hardware Architecture

Fixed-Point Quantization

HDL Coding

Implementation

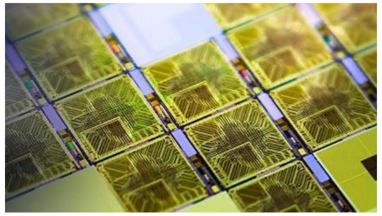
Infrastructure Creation

Component Development

Simulation and Debug

Coverage

Prototype



Production

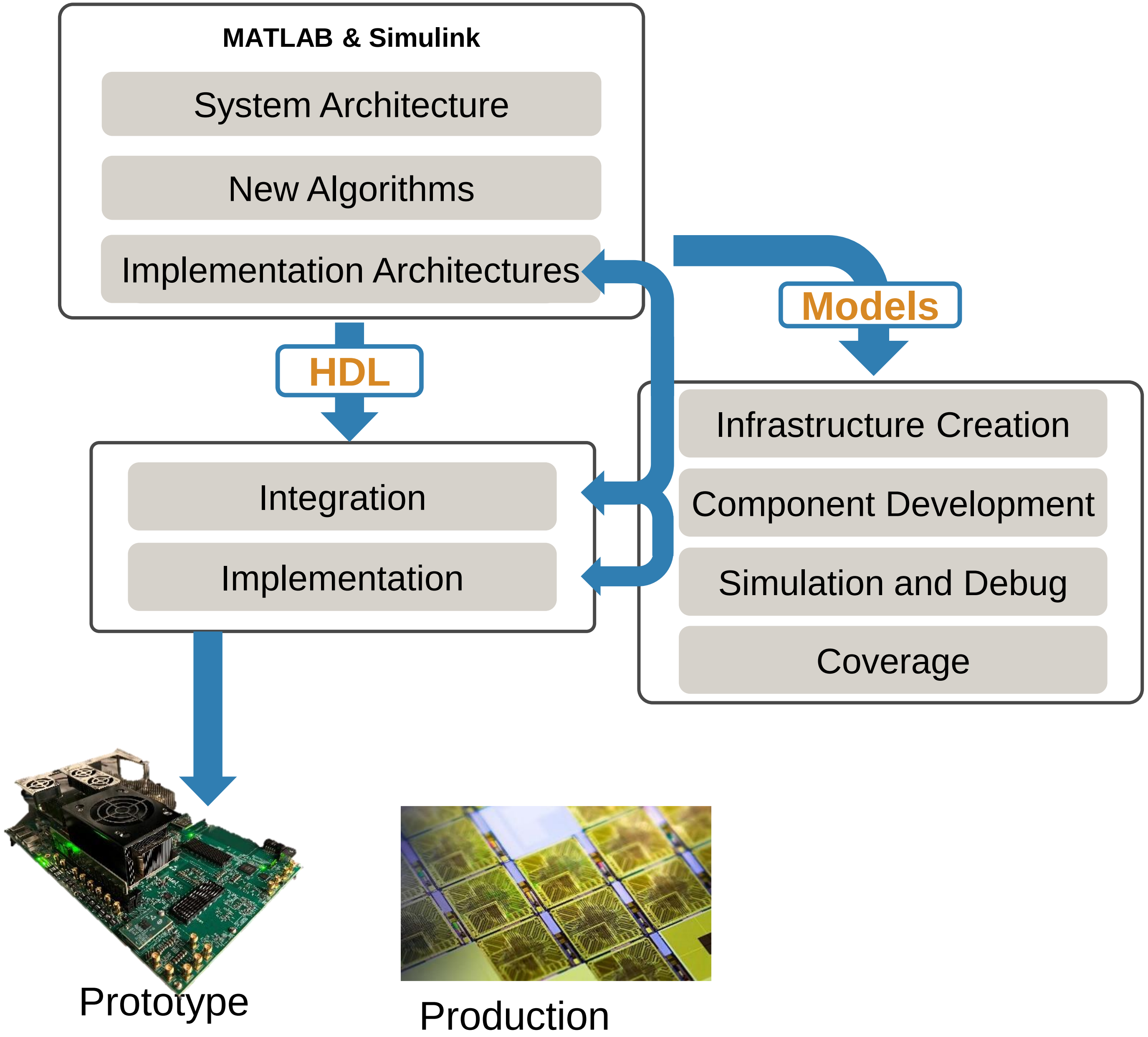
Collaborate to Innovate at a High-Level

Collaborate across teams to explore implementation options

Verify high-abstraction models in the system context

Generate production-quality HDL and verification models

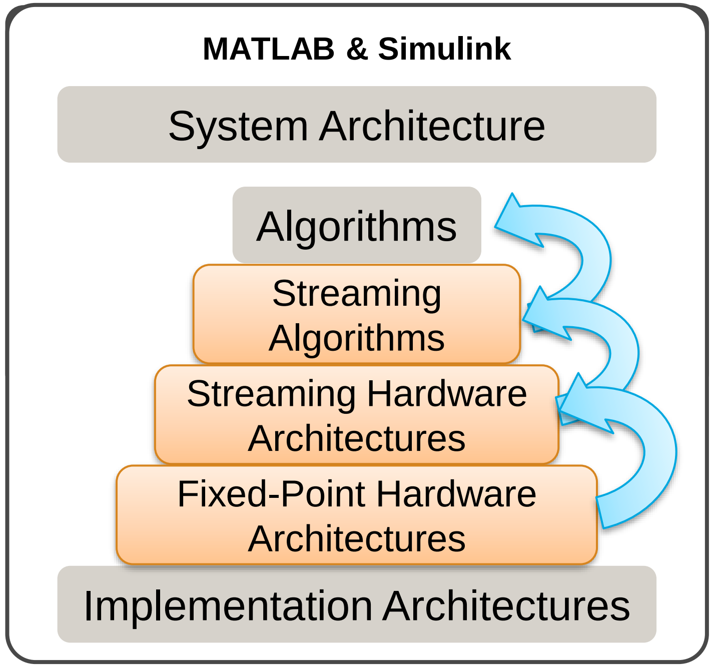
Target and debug FPGA prototype boards



General Approach: Use the Strengths of MATLAB and Simulink

MATLAB

- ✓ Large data sets
- ✓ Explore mathematics
- ✓ Control logic
- ✓ Data visualization



MATLAB & Simulink

System Architecture

Algorithms

Streaming Algorithms

Streaming Hardware Architectures

Fixed-Point Hardware Architectures

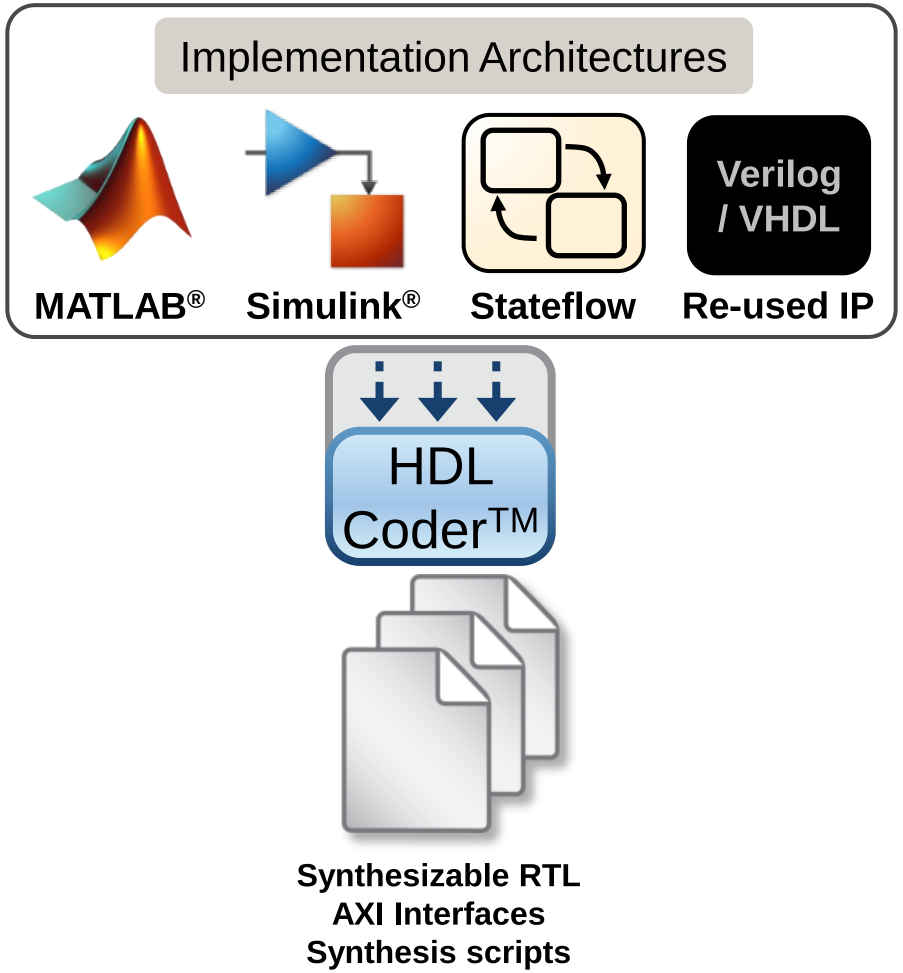
Implementation Architectures

Simulink

- ✓ Parallel architectures
- ✓ Timing
- ✓ Data type propagation
- ✓ Mixed-signal modeling

HDL Code Generation

Connect system/algorithm design to FPGA/ASIC hardware



Implementation Architectures

MATLAB® Simulink® Stateflow Re-used IP

Verilog / VHDL

HDL Coder™

Synthesizable RTL
AXI Interfaces
Synthesis scripts

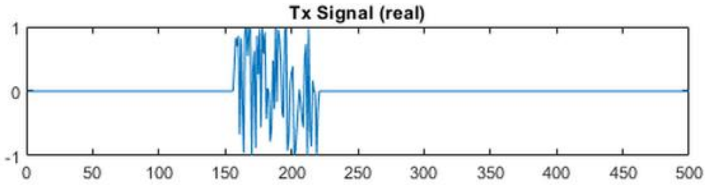
- Generate readable, traceable Verilog/VHDL
  - Prototype: automatically target popular boards and kits
  - Production: generate IP core with AXI interfaces
- Quickly adapt to changes and re-generate
- Automatically convert to fixed-point or use native floating point
- Customize automatic optimizations and code settings
- Re-use for different targets, different project goals

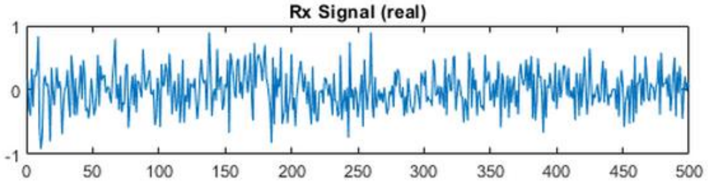
Example: Pulse Detector

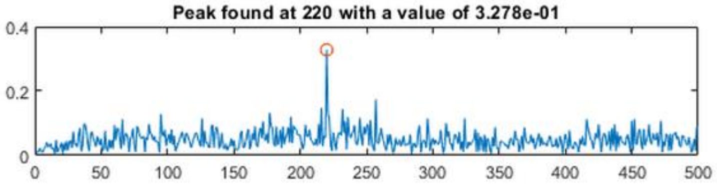
Send

Receive

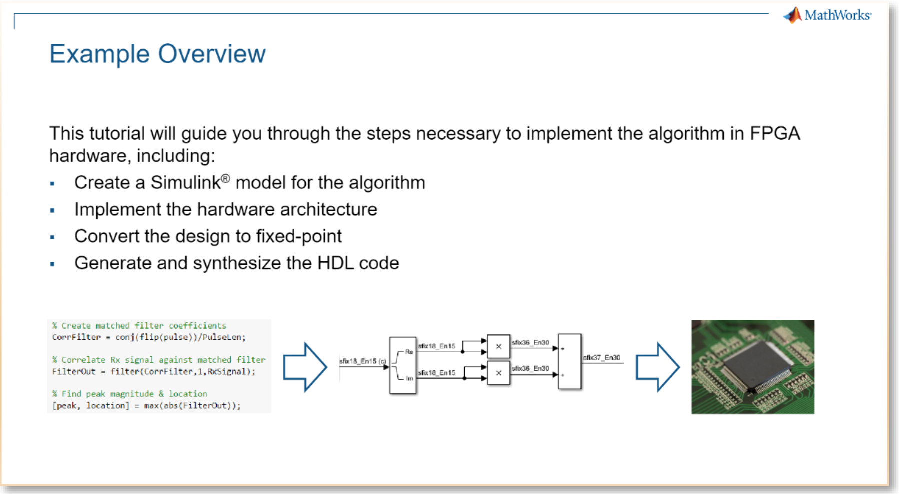
Detect







[HDL Coder Self-Guided Tutorial]



Example Overview

This tutorial will guide you through the steps necessary to implement the algorithm in FPGA hardware, including:

- Create a Simulink® model for the algorithm
- Implement the hardware architecture
- Convert the design to fixed-point
- Generate and synthesize the HDL code

```
% Create matched filter coefficients
CorrFilter = conj(flip(pulse))/PulseLen;

% Correlate Rx signal against matched filter
FilterOut = filter(CorrFilter,1,RxSignal);

% Find peak magnitude & location
[peak, location] = max(abs(FilterOut));
```

Achieve results with less manual effort

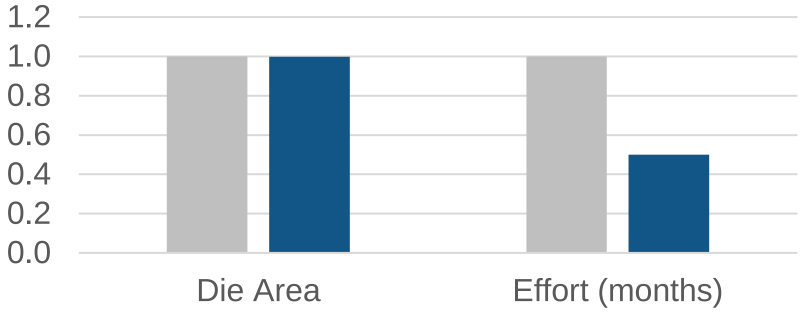
All numbers normalized to 1.0 for “Manual coding”
Smaller numbers are better

▨ Manual coding

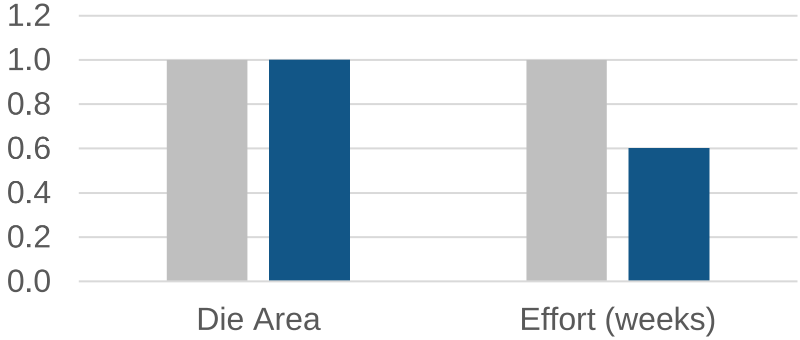
■ Simulink + HDL Coder

Qualcomm India

Wide-band chain from front-end receiver ASIC

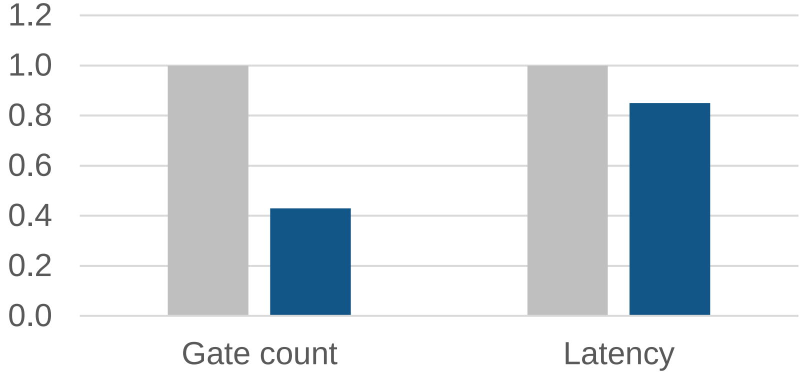


Decimation FIR filter ASIC



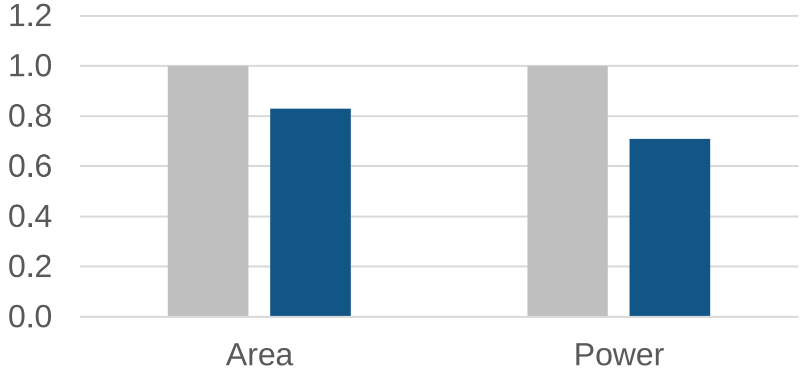
Faraday

Flash NAND Controller ASIC



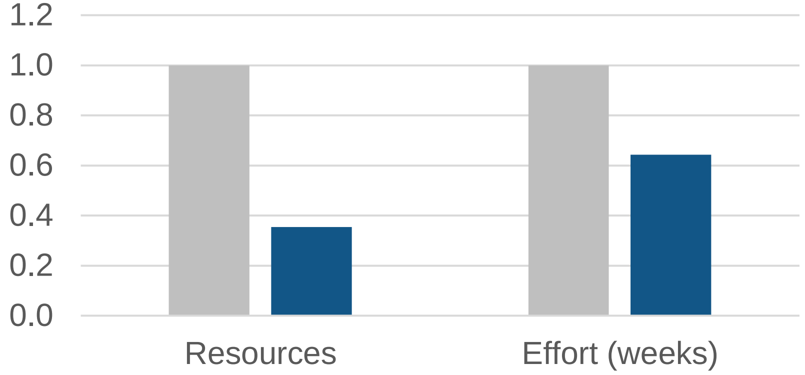
Bosch

Automotive ECU ASIC



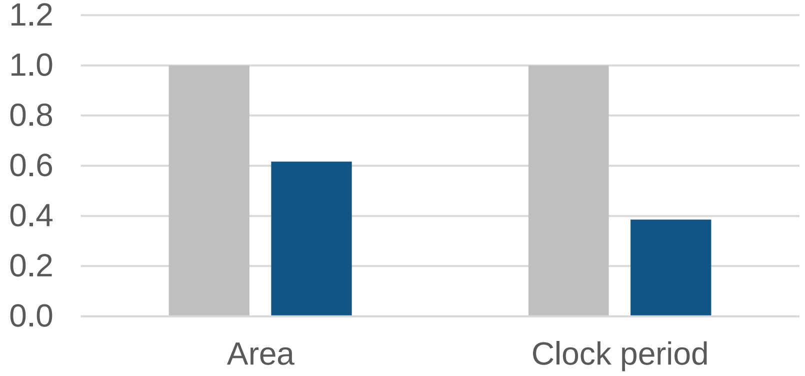
IFM Engineering

3D time-of-flight camera FPGA



Nokia

FPGA prototype of an ASIC



Application-Specific Solutions Extend HDL Coder

Wireless: 5G, LTE, WLAN, Satcom
Wireless HDL Toolbox

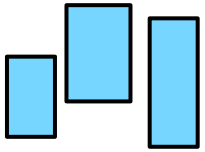
Controls: Motor, power electronics, battery management
HDL Coder / Embedded Coder / Motor Control Blockset / SoC Blockset

Signal processing / Radar / LIDAR
DSP HDL Toolbox

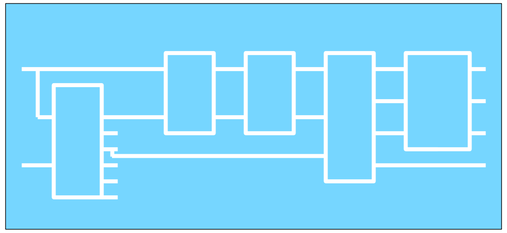
Vision: Automated driving, inspection, surveillance, medical imaging
Vision HDL Toolbox

AI: deep learning, machine learning
Deep Learning HDL Toolbox

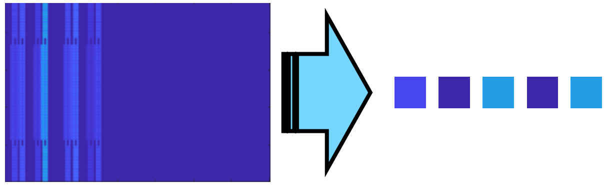
Build using HDL-optimized blocks



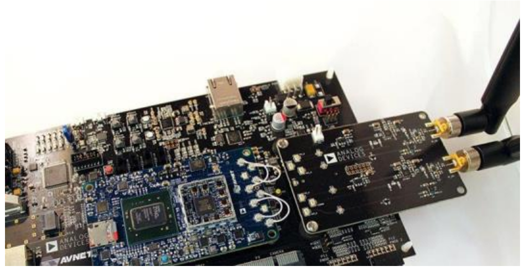
Integrate pre-built subsystems



Manage data with utilities

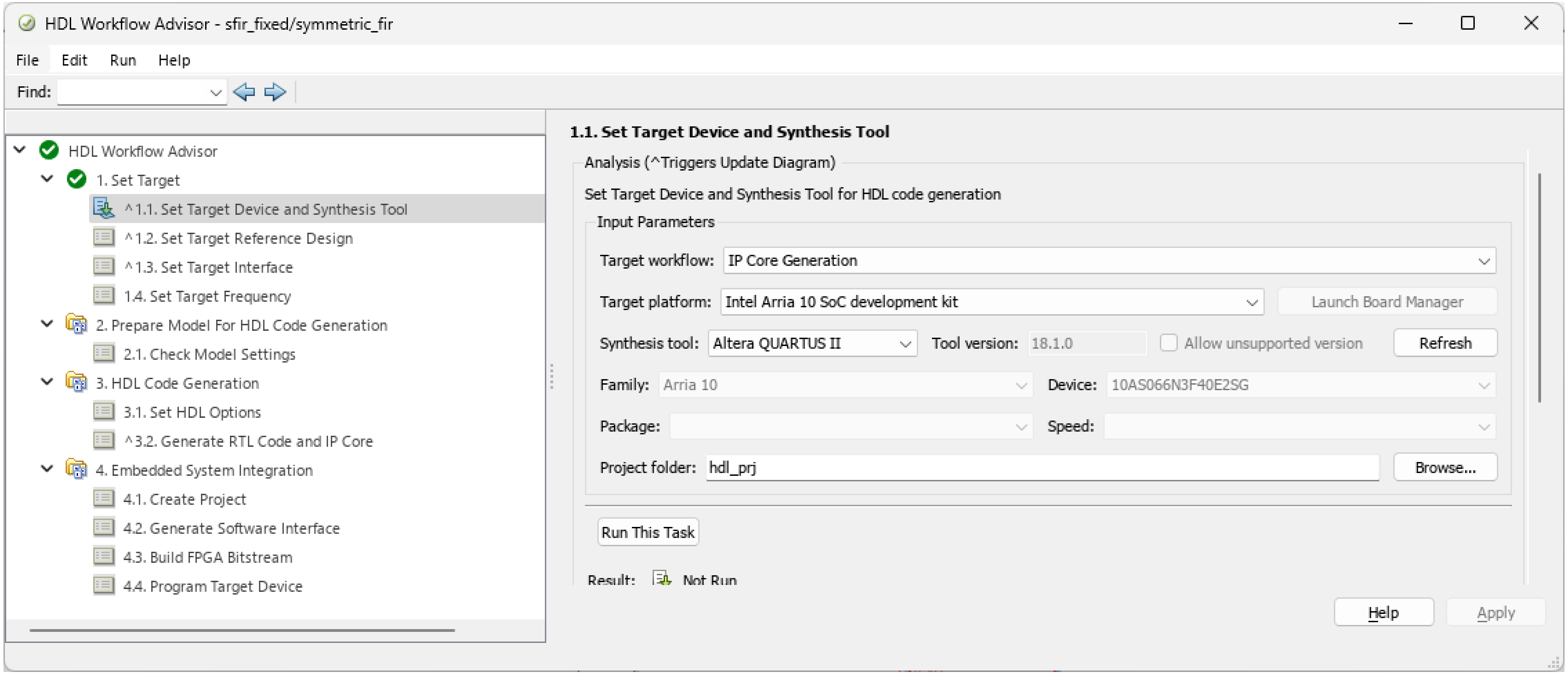


Directly target FPGA-based kits



Support for Intel Development Boards from HDL Product Family

- HDL Coder
  - Intel Development Board Support from HDL Coder
  - Intel SoC FPGA Support from HDL Coder
- Embedded Coder - Intel SoC FPGA Support from Embedded Coder
- HDL Verifier - Intel FPGA Board Support from HDL Verifier
- SoC Blockset - Intel Support from SoC Blockset
- Deep Learning HDL Toolbox - Intel FPGA and SoC Support from Deep Learning HDL Toolbox


HDL Workflow Advisor - sfir_fixed/symmetric_fir
File
Edit
Run
Help
Find:
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 Model Settings
3. HDL Code Generation
3.1. Set HDL Options
^3.2. Generate RTL Code and IP Core
4. Embedded System Integration
4.1. Create Project
4.2. Generate Software Interface
4.3. Build FPGA Bitstream
4.4. Program Target Device
Launch Board Manager
Refresh
Browse...
Run This Task
Help
Apply

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: Intel Arria 10 SoC development kit

Synthesis tool: Altera QUARTUS II

Tool version: 18.1.0

□ Allow unsupported version

Family: Arria 10

Device: 10AS066N3F40E2SG

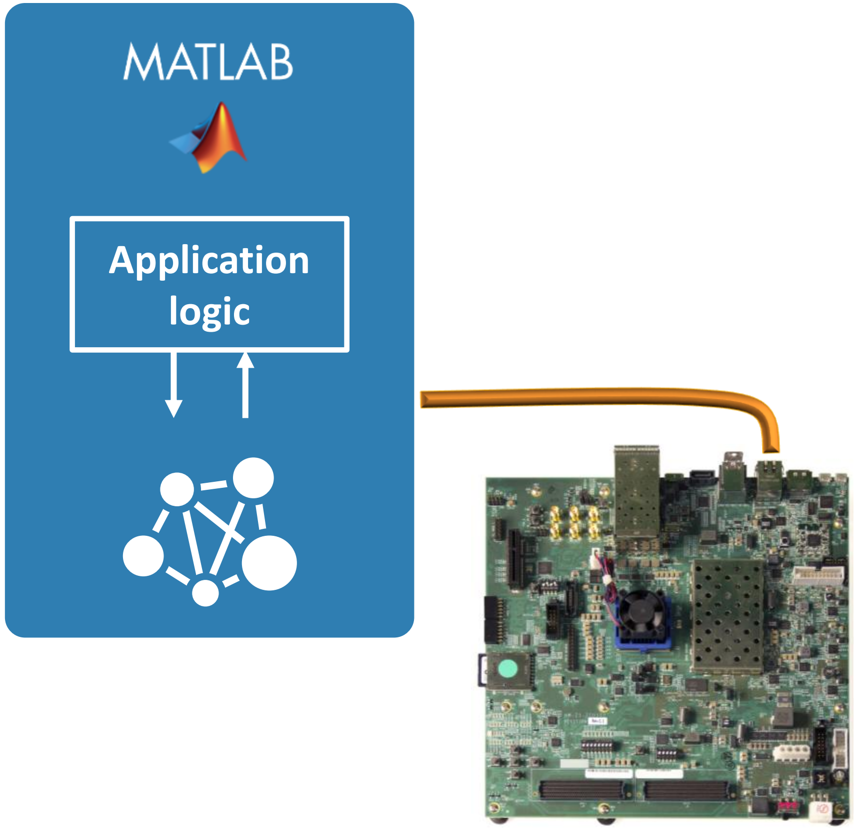
Package:

Speed:

Project folder: hdl_prj

Result: Not Run

Deep Learning HDL Toolbox: Prototype & Deploy DL Networks



MATLAB

Application logic

Deep Learning HDL Toolbox^{TM}

- Prototype network on FPGA
- Assess memory usage, latency, and accuracy
  - Adjust network and iterate
  - Quantize to fixed-point
- Generate customized deep learning processor HDL

…all from within MATLAB!

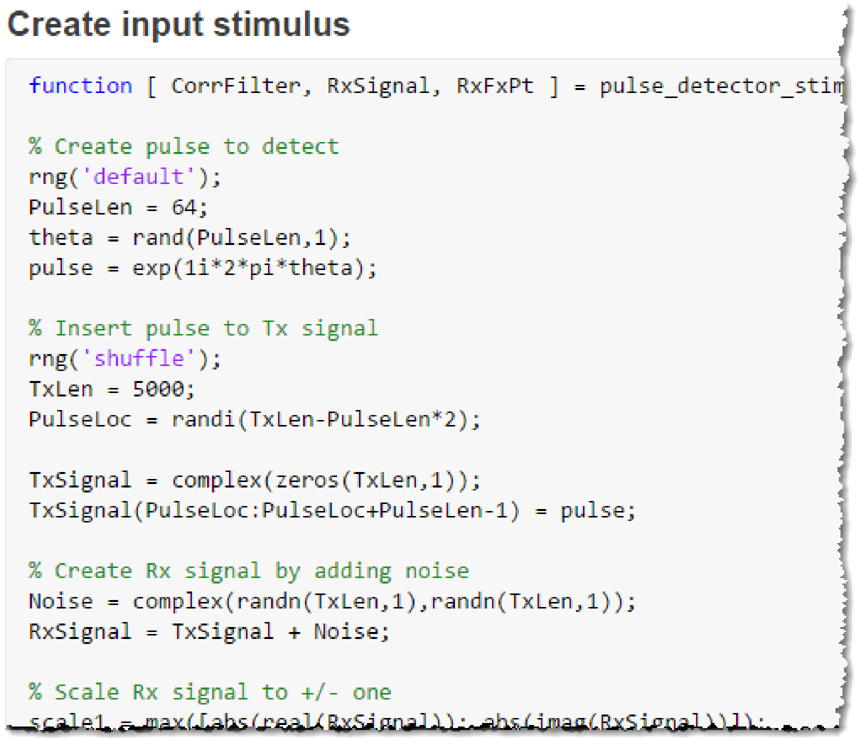
Partition Hardware-Targeted Design, System Context, Testbench

Algorithm Stimulus

Hardware Algorithm

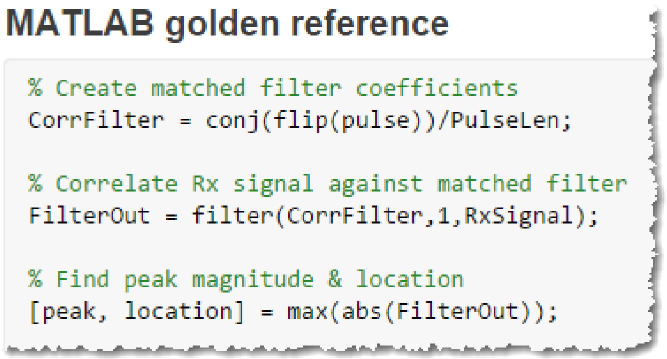
Software Algorithm

Analysis



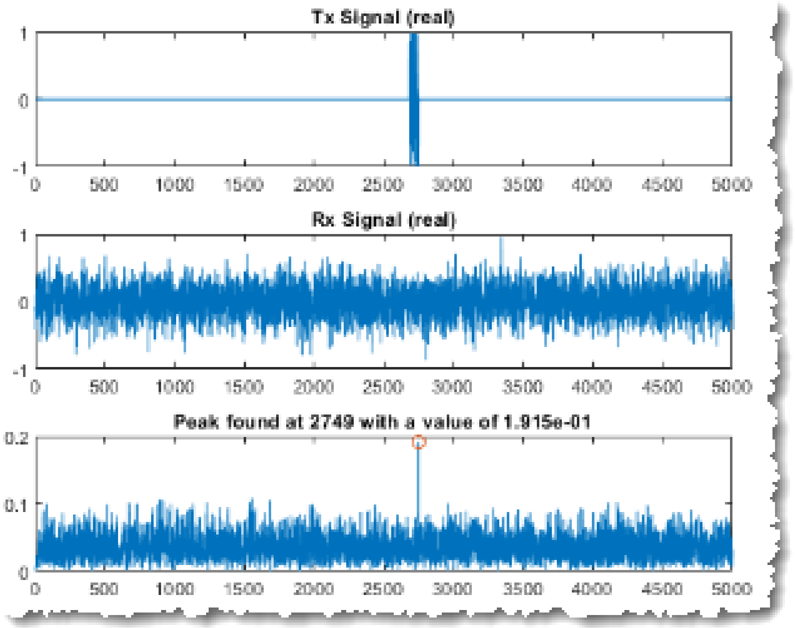
Create input stimulus

```
function [ CorrFilter, RxSignal, RxFxPt ] = pulse_detector_stim

% Create pulse to detect
rng('default');
PulseLen = 64;
theta = rand(PulseLen,1);
pulse = exp(1i*2*pi*theta);

% Insert pulse to Tx signal
rng('shuffle');
TxLen = 5000;
PulseLoc = randi(TxLen-PulseLen*2);

TxSignal = complex(zeros(TxLen,1));
TxSignal(PulseLoc:PulseLoc+PulseLen-1) = pulse;

% Create Rx signal by adding noise
Noise = complex(randn(TxLen,1),randn(TxLen,1));
RxSignal = TxSignal + Noise;

% Scale Rx signal to +/- one
scale1 = max([abs(real(RxSignal)); abs(imag(RxSignal))]);
```

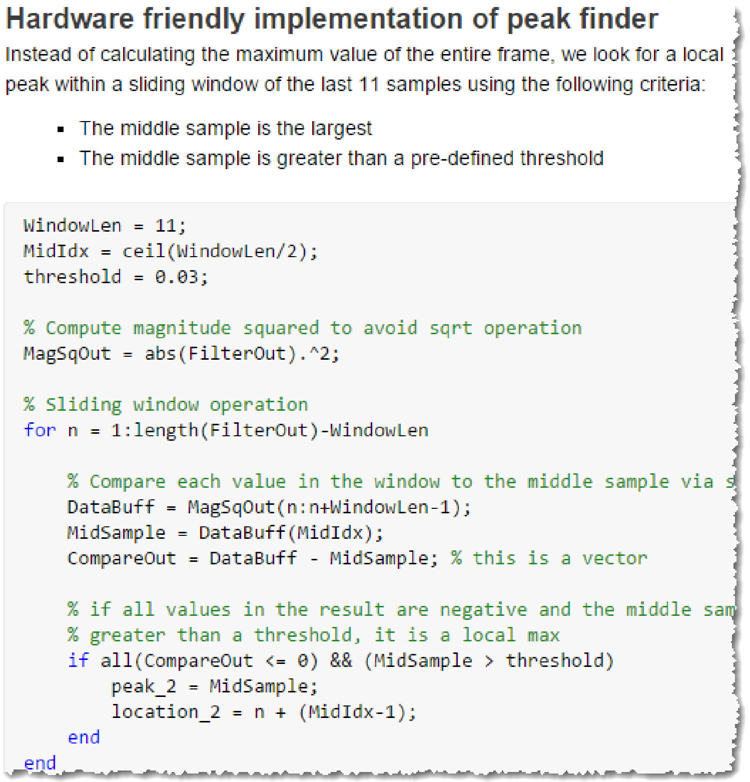


MATLAB golden reference

```
% Create matched filter coefficients
CorrFilter = conj(flip(pulse))/PulseLen;

% Correlate Rx signal against matched filter
FilterOut = filter(CorrFilter,1,RxSignal);

% Find peak magnitude & location
[peak, location] = max(abs(FilterOut));
```



Streaming Algorithms: MATLAB or Simulink…or Both

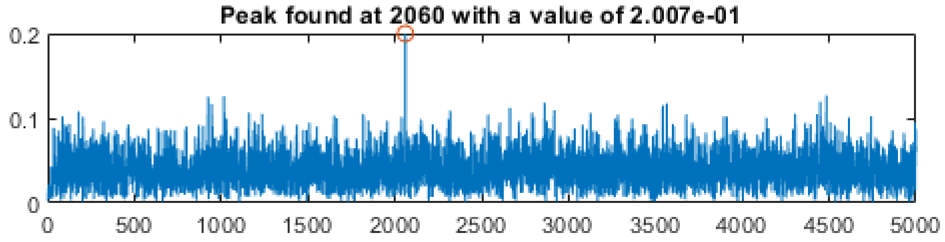


Hardware friendly implementation of peak finder

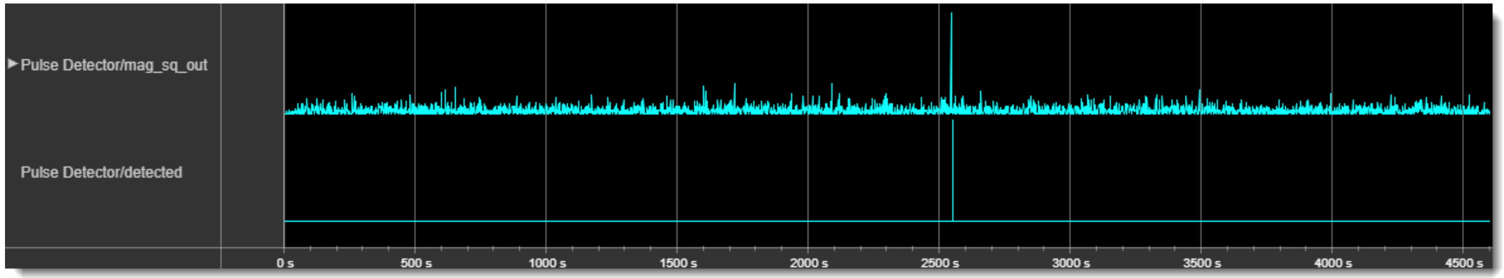
Instead of calculating the maximum value of the entire frame, we look for a local peak within a sliding window of the last 11 samples using the following criteria:

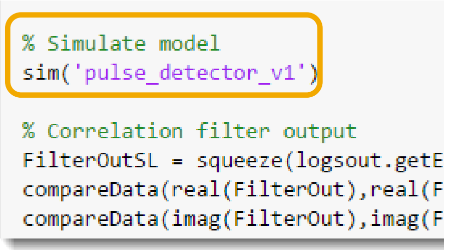
- The middle sample is the largest
- The middle sample is greater than a pre-defined threshold

```
WindowLen = 11;
MidIdx = ceil(WindowLen/2);
threshold = 0.03;

% Compute magnitude squared to avoid sqrt operation
MagSqOut = abs(FilterOut).^2;

% Sliding window operation
for n = 1:length(FilterOut)-WindowLen

    % Compare each value in the window to the middle sample via s
    DataBuff = MagSqOut(n:n+WindowLen-1);
    MidSample = DataBuff(MidIdx);
    CompareOut = DataBuff - MidSample; % this is a vector

    % if all values in the result are negative and the middle sam
    % greater than a threshold, it is a local max
    if all(CompareOut <= 0) && (MidSample > threshold)
        peak_2 = MidSample;
        location_2 = n + (MidIdx-1);
    end
end
```

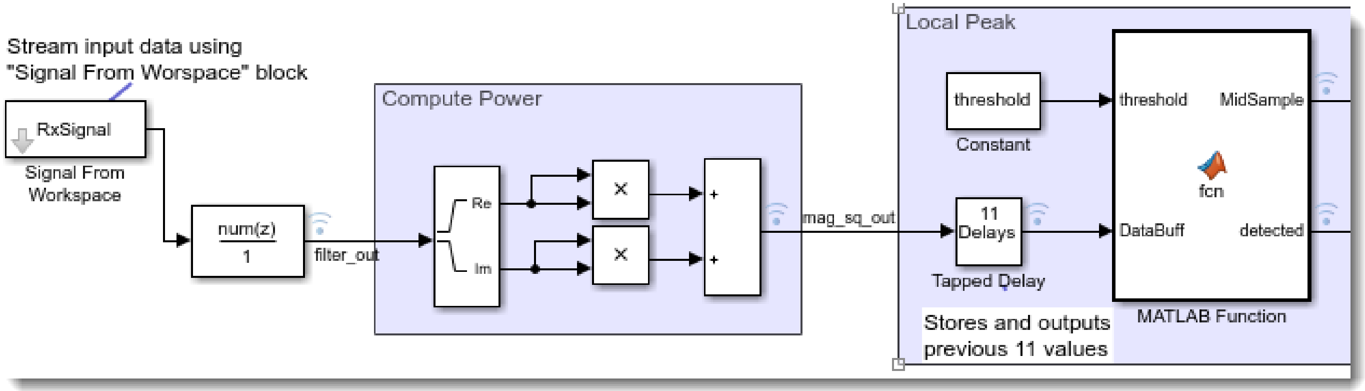


Peak found at 2060 with a value of 2.007e-01





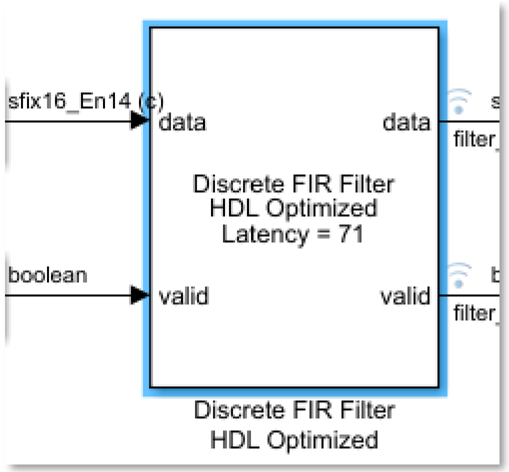
```
% Simulate model
sim('pulse_detector_v1')

% Correlation filter output
FilterOutSL = squeeze(logsout.getE
compareData(real(FilterOut),real(F
compareData(imag(FilterOut),imag(F
```

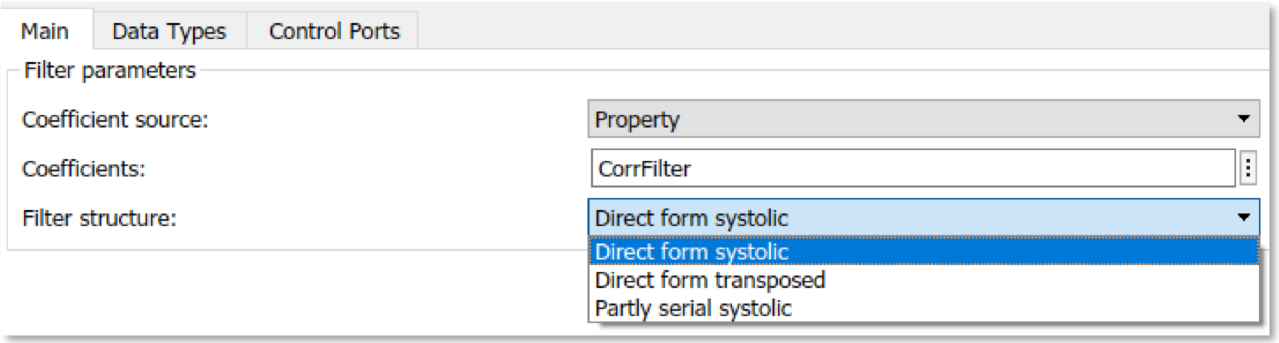


Stream input data using "Signal From Worspace" block

Stores and outputs previous 11 values

Hardware Architectures: Design Using >300 HDL-Ready Blocks




Main | Data Types | Control Ports

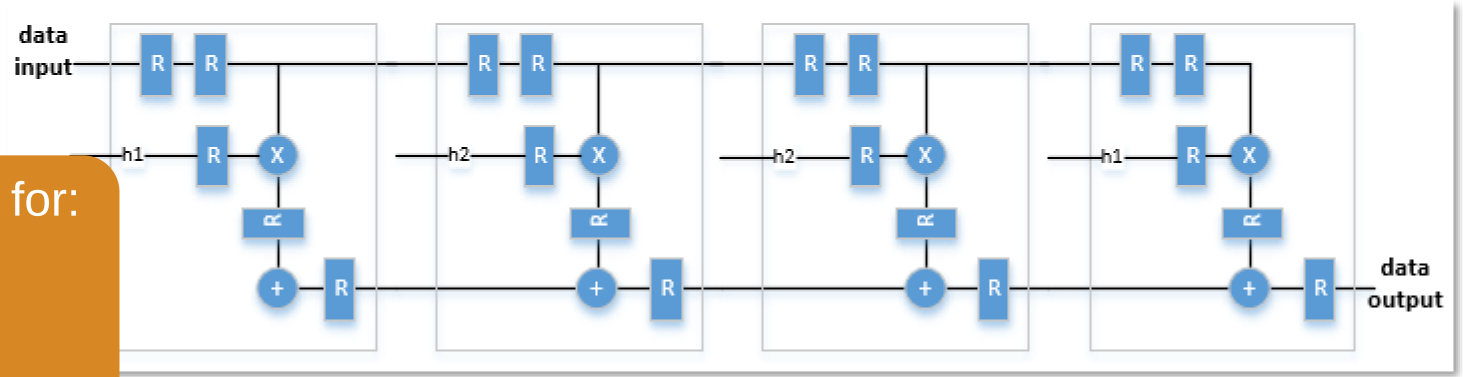
Filter parameters

Coefficient source: Property

Coefficients: CorrFilter

Filter structure: Direct form systolic

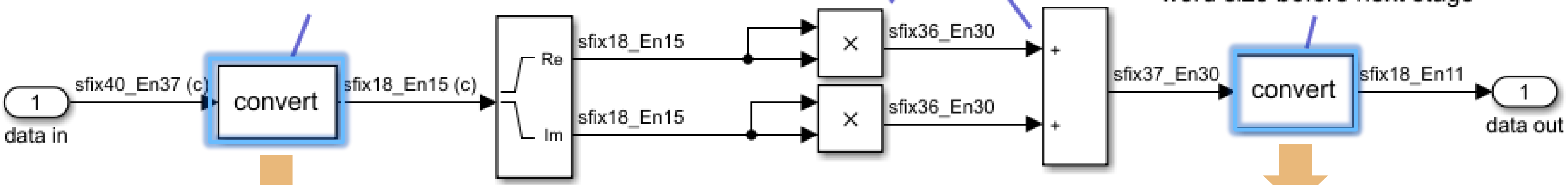
- Direct form systolic
- Direct form transposed
- Partly serial systolic



Application-specific IP available for:
DSP
Wireless
Linear Algebra
AI/Deep Learning
Video/Image Processing

Fixed-Point: Analyze Range and Precision

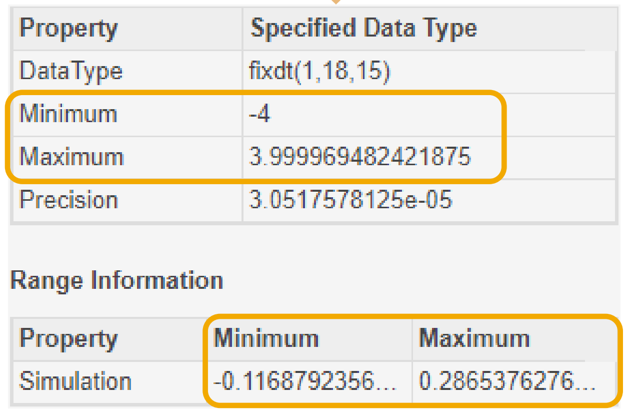
- Automatic range collection with overflow logging
- Maximize precision of chosen word lengths using simulation data



Reduce multiplier input word size to 18-bit

Use full-precision for DSP block mapping

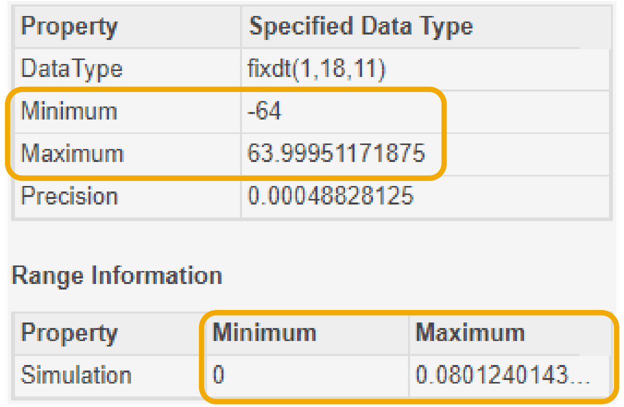
Reduce mag-squared output word size before next stage



Property	Specified Data Type
DataType	fixdt(1,18,15)
Minimum	-4
Maximum	3.999969482421875
Precision	3.0517578125e-05

Range Information

Property	Minimum	Maximum
Simulation	-0.1168792356...	0.2865376276...

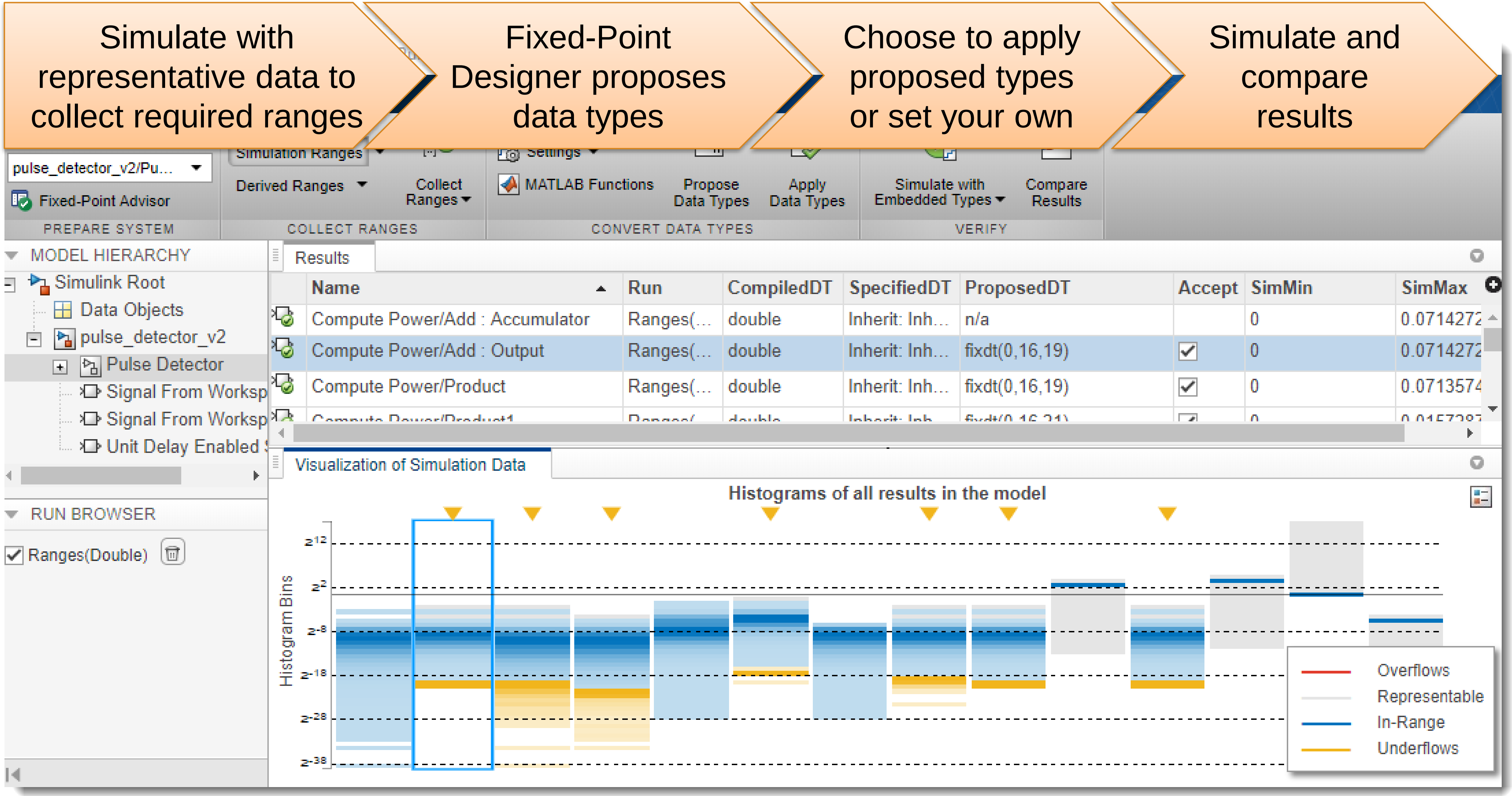


Property	Specified Data Type
DataType	fixdt(1,18,11)
Minimum	-64
Maximum	63.99951171875
Precision	0.00048828125

Range Information

Property	Minimum	Maximum
Simulation	0	0.0801240143...

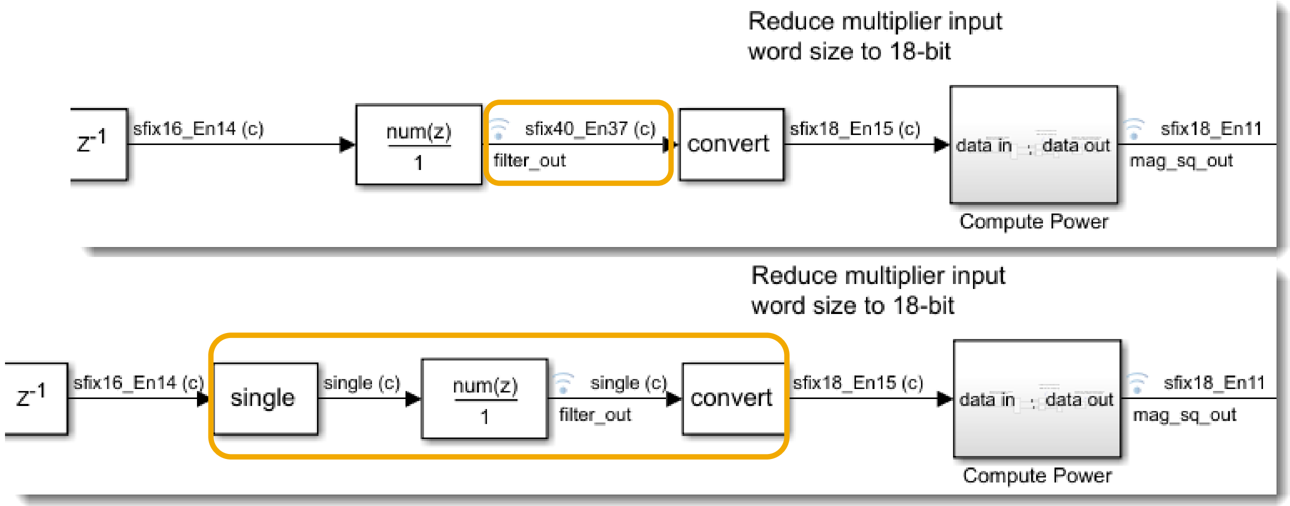
Fixed-Point: Automated Approach



Simulate with representative data to collect required ranges → Fixed-Point Designer proposes data types → Choose to apply proposed types or set your own → Simulate and compare results

Name	Run	CompiledDT	SpecifiedDT	ProposedDT	Accept	SimMin	SimMax
Compute Power/Add : Accumulator	Ranges(...	double	Inherit: Inh...	n/a		0	0.0714272
Compute Power/Add : Output	Ranges(...	double	Inherit: Inh...	fixdt(0,16,19)	☑	0	0.0714272
Compute Power/Product	Ranges(...	double	Inherit: Inh...	fixdt(0,16,19)	☑	0	0.0713574
Compute Power/Product1	Ranges(...	double	Inherit: Inh...	fixdt(0,16,21)	☑	0	0.0157287

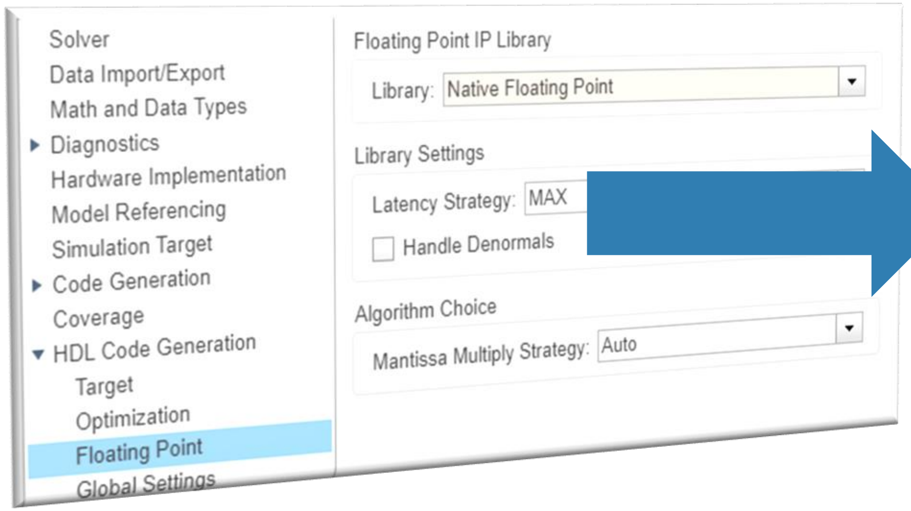
Sometimes It’s More Efficient to Generate Floating Point Hardware



HDL Coder™ Native Floating Point

- Extensive math and trigonometric operator support
- Optimal implementations without sacrificing numerical accuracy
- Mix floating- and fixed-point operations
- Generate target-independent HDL

embeddedaward2017



Floating Point IP Library

Library: Native Floating Point

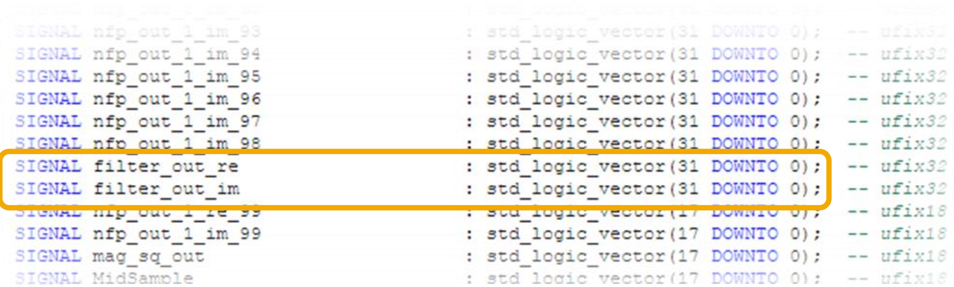
Library Settings

Latency Strategy: MAX

☐ Handle Denormals

Algorithm Choice

Mantissa Multiply Strategy: Auto



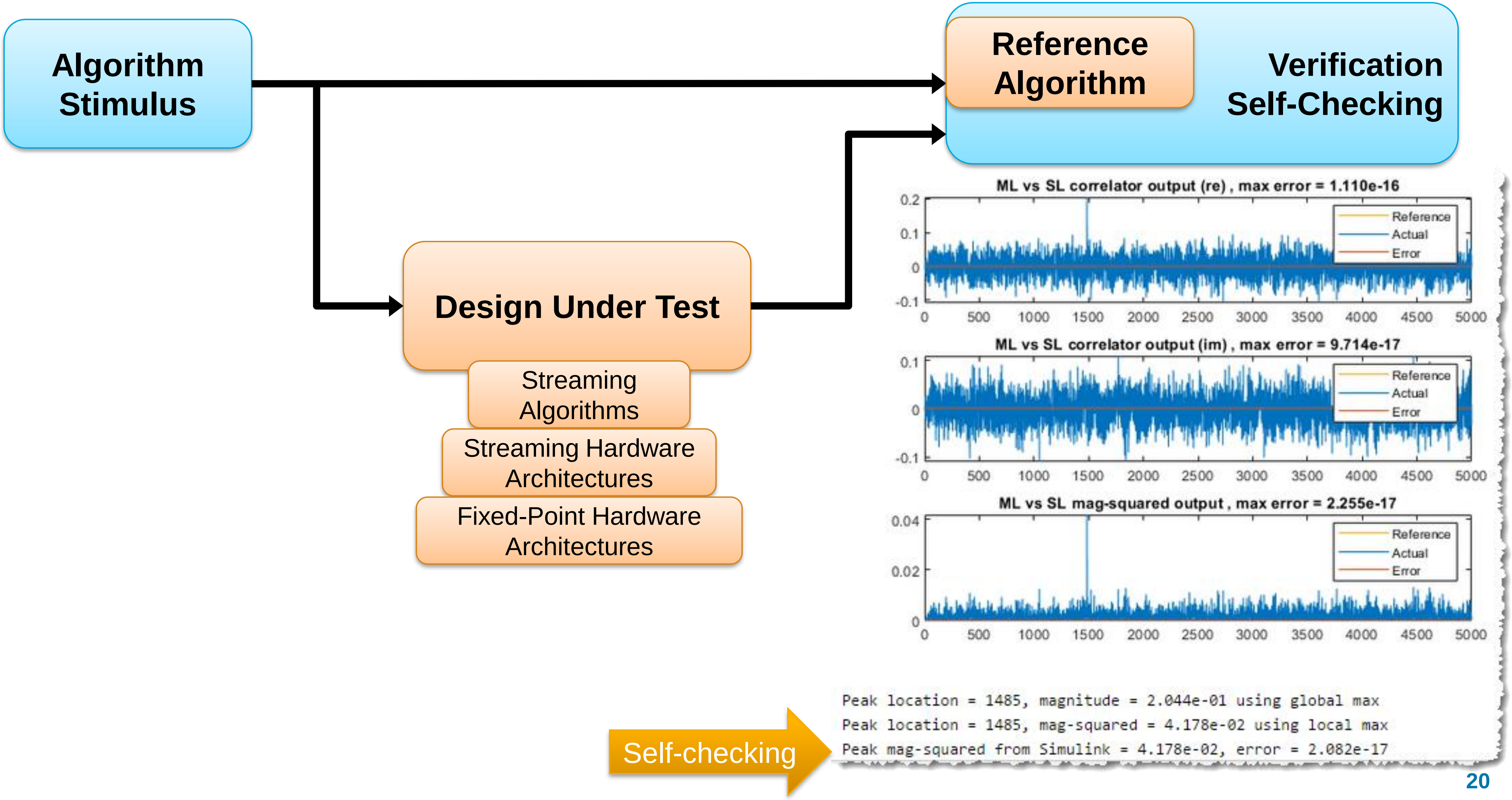
```
SIGNAL nfp_out_1_im_93                        : std_logic_vector(31 DOWNTO 0);  -- ufix32
SIGNAL nfp_out_1_im_94                        : std_logic_vector(31 DOWNTO 0);  -- ufix32
SIGNAL nfp_out_1_im_95                        : std_logic_vector(31 DOWNTO 0);  -- ufix32
SIGNAL nfp_out_1_im_96                        : std_logic_vector(31 DOWNTO 0);  -- ufix32
SIGNAL nfp_out_1_im_97                        : std_logic_vector(31 DOWNTO 0);  -- ufix32
SIGNAL nfp_out_1_im_98                        : std_logic_vector(31 DOWNTO 0);  -- ufix32
SIGNAL filter_out_re                          : std_logic_vector(31 DOWNTO 0);  -- ufix32
SIGNAL filter_out_im                          : std_logic_vector(31 DOWNTO 0);  -- ufix32
SIGNAL nfp_out_1_re_99                        : std_logic_vector(17 DOWNTO 0);  -- ufix18
SIGNAL nfp_out_1_im_99                        : std_logic_vector(17 DOWNTO 0);  -- ufix18
SIGNAL mag_sq_out                             : std_logic_vector(17 DOWNTO 0);  -- ufix18
SIGNAL MidSample                              : std_logic_vector(17 DOWNTO 0);  -- ufix18
```

	Fixed point	Floating point
LUTs	10k	25k
DSP slices	50	100
Development time	~1 week	~1 day

~2x more resources

~5x less development effort

Verify Implementation Decisions Against Golden Reference



Algorithm Stimulus

Design Under Test

Streaming Algorithms

Streaming Hardware Architectures

Fixed-Point Hardware Architectures

Reference Algorithm

Verification Self-Checking

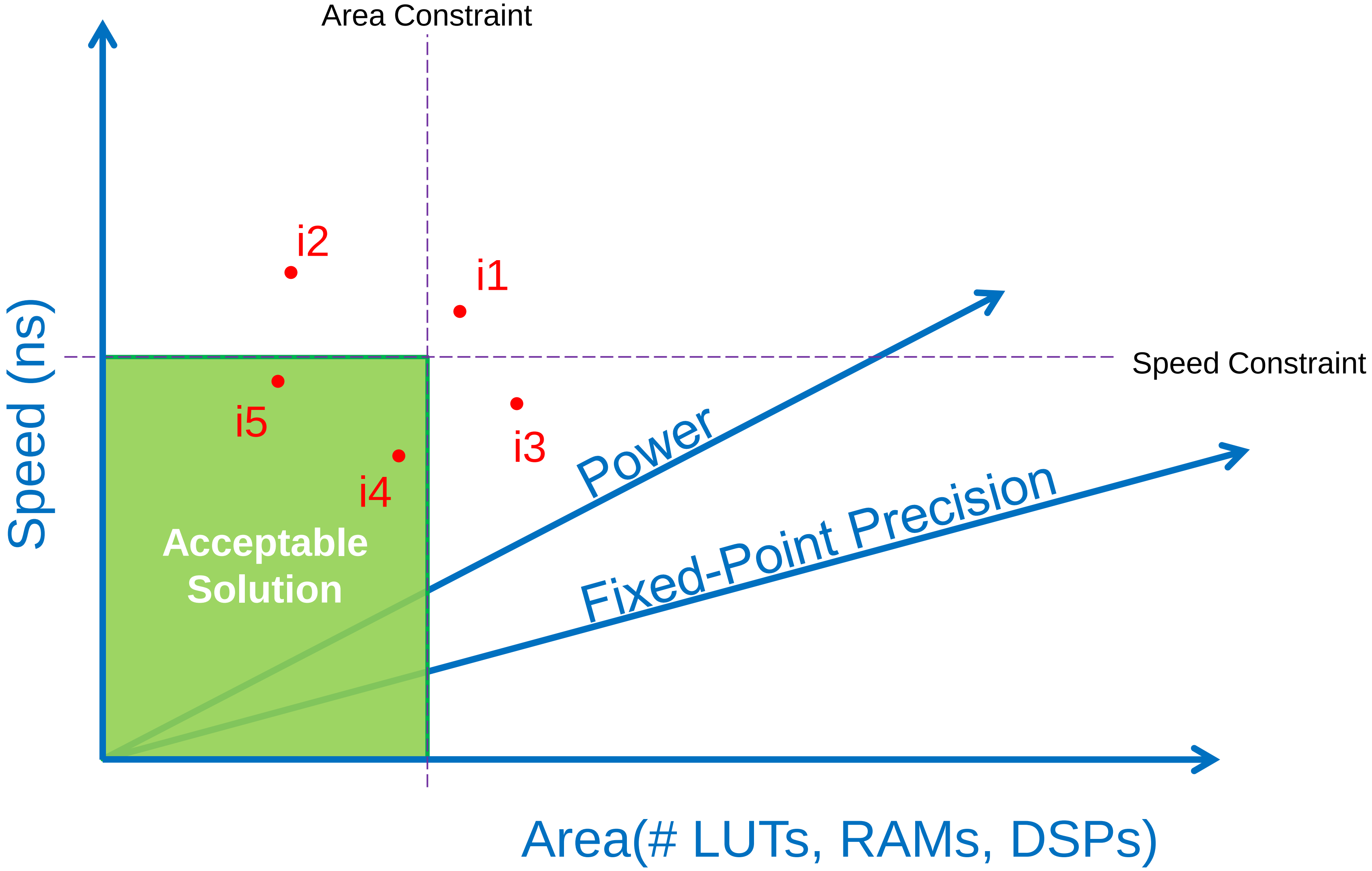
Self-checking

ML vs SL correlator output (re) , max error = 1.110e-16
0.2
0.1
0
-0.1
0
500
1000
1500
2000
2500
3000
3500
4000
4500
5000
Reference
Actual
Error
ML vs SL correlator output (im) , max error = 9.714e-17
0.1
0
-0.1
Reference
Actual
Error
ML vs SL mag-squared output , max error = 2.255e-17
0.04
0.02
0
Reference
Actual
Error

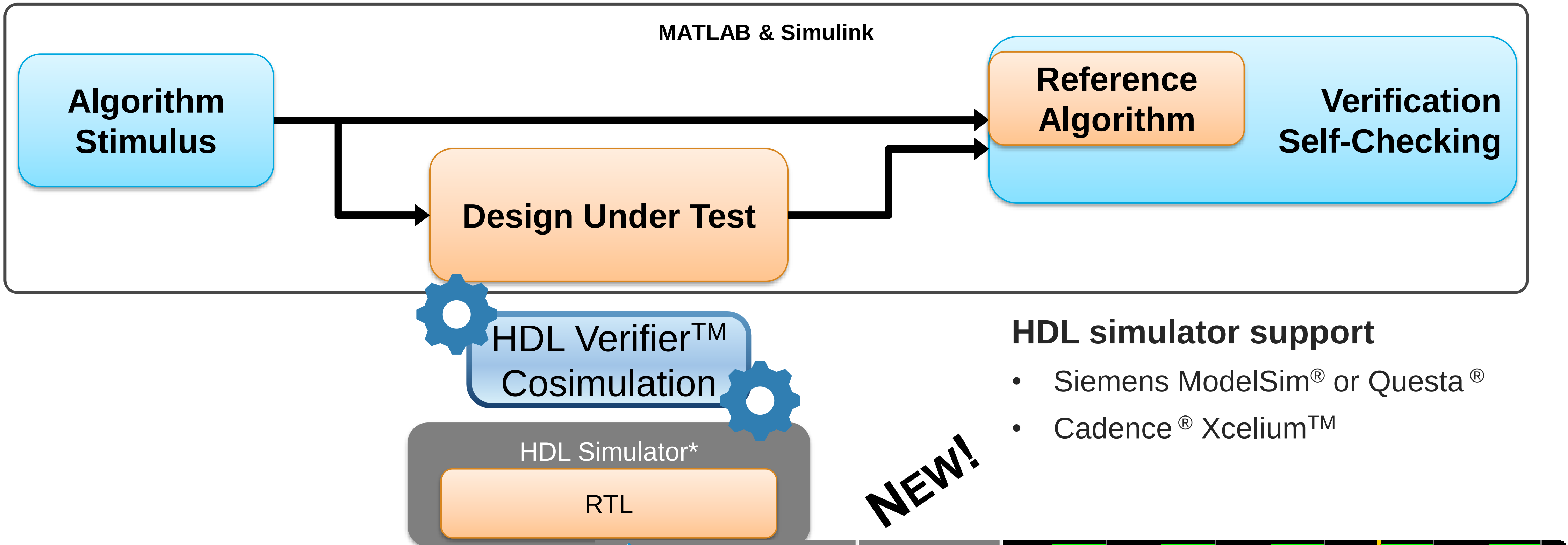
```
Peak location = 1485, magnitude = 2.044e-01 using global max
Peak location = 1485, mag-squared = 4.178e-02 using local max
Peak mag-squared from Simulink = 4.178e-02, error = 2.082e-17
```

Explore a Broad Range of Hardware Architectures

Automatically Set or Fully Control



Verify HDL/FPGA Implementation

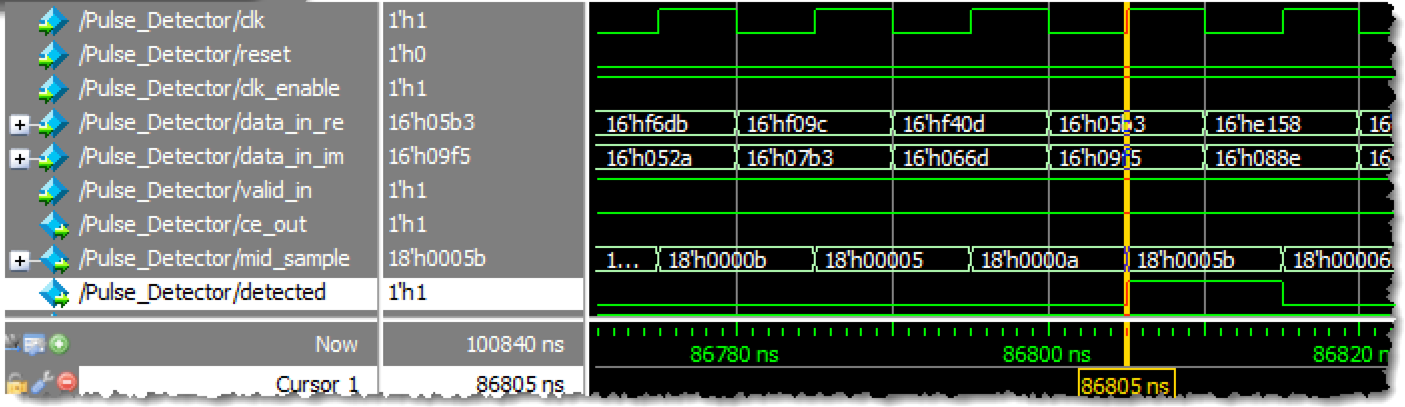


HDL simulator support

- Siemens ModelSim® or Questa ®
- Cadence ® Xcelium™

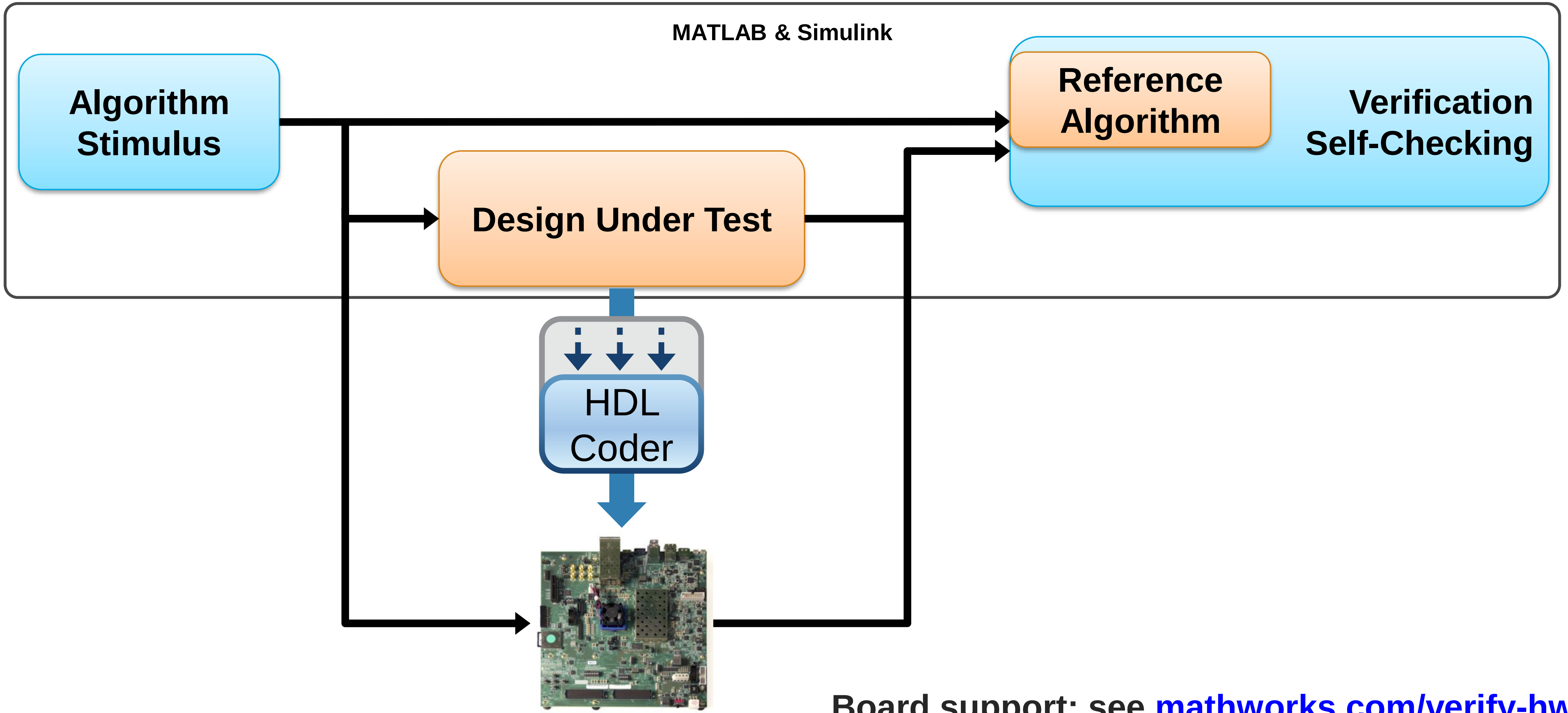
NEW!

- Reuse MATLAB/Simulink test environment
- Generate co-simulation infrastructure and handshaking
- Analyze both the design and test environment



```
/Pulse_Detector/clk            1'h1
/Pulse_Detector/reset          1'h0
/Pulse_Detector/clk_enable     1'h1
/Pulse_Detector/data_in_re     16'h05b3
/Pulse_Detector/data_in_im     16'h09f5
/Pulse_Detector/valid_in       1'h1
/Pulse_Detector/ce_out         1'h1
/Pulse_Detector/mid_sample     18'h0005b
/Pulse_Detector/detected       1'h1
                     Now       100840 ns
                Cursor 1        86805 ns
```

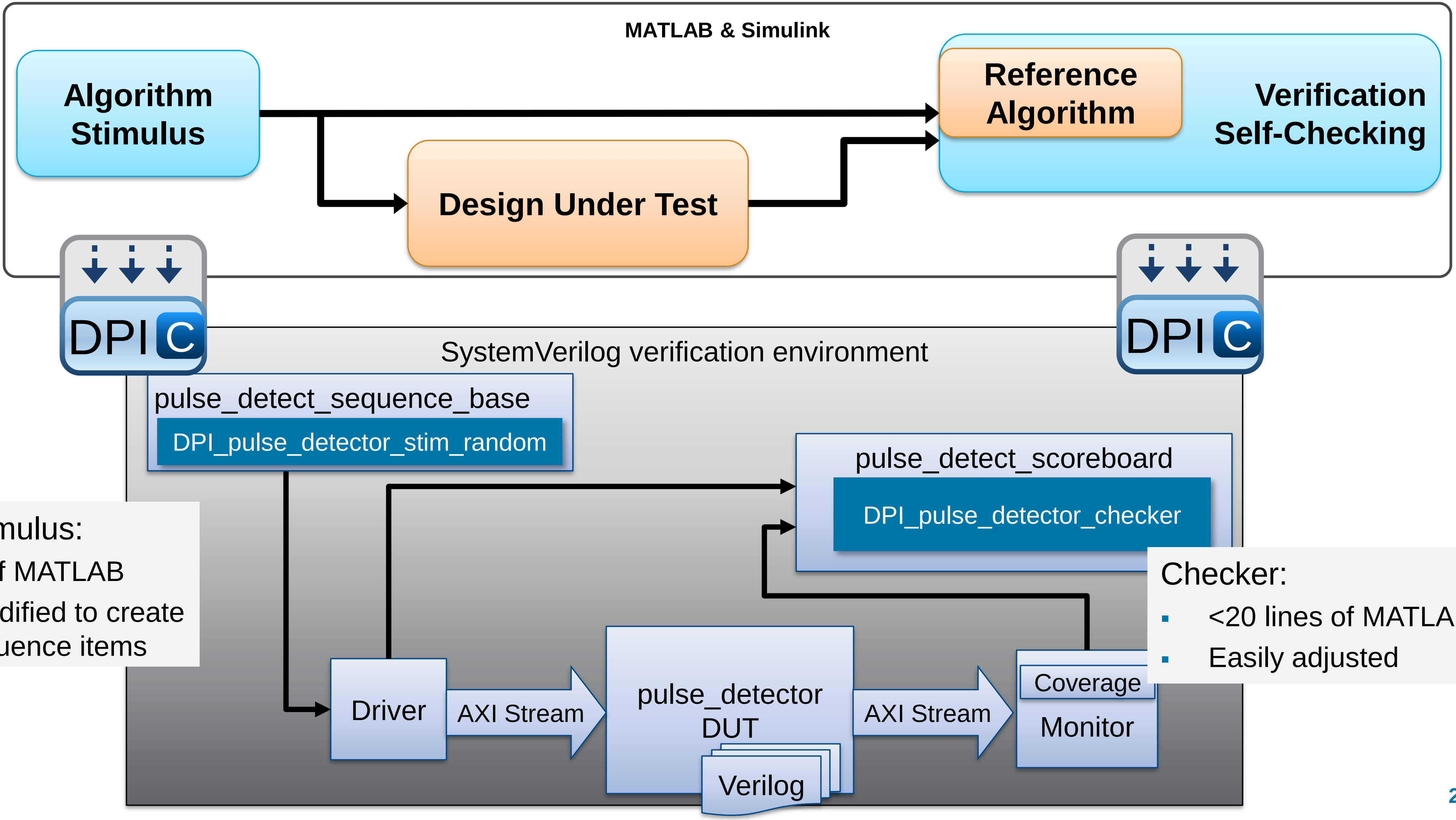
Prototype and Debug on FPGA Hardware



Board support: see mathworks.com/verify-hw

- Options:
  - FPGA-in-the-loop with MATLAB/Simulink test environment
  - Deploy to FPGA with test point data capture
  - Interactively stimulate FPGA from MATLAB
- Debug actual hardware implementation directly from MATLAB and Simulink

Generate Models for Production Verification Environment



Random stimulus:

- 15 lines of MATLAB
- Easily modified to create more sequence items

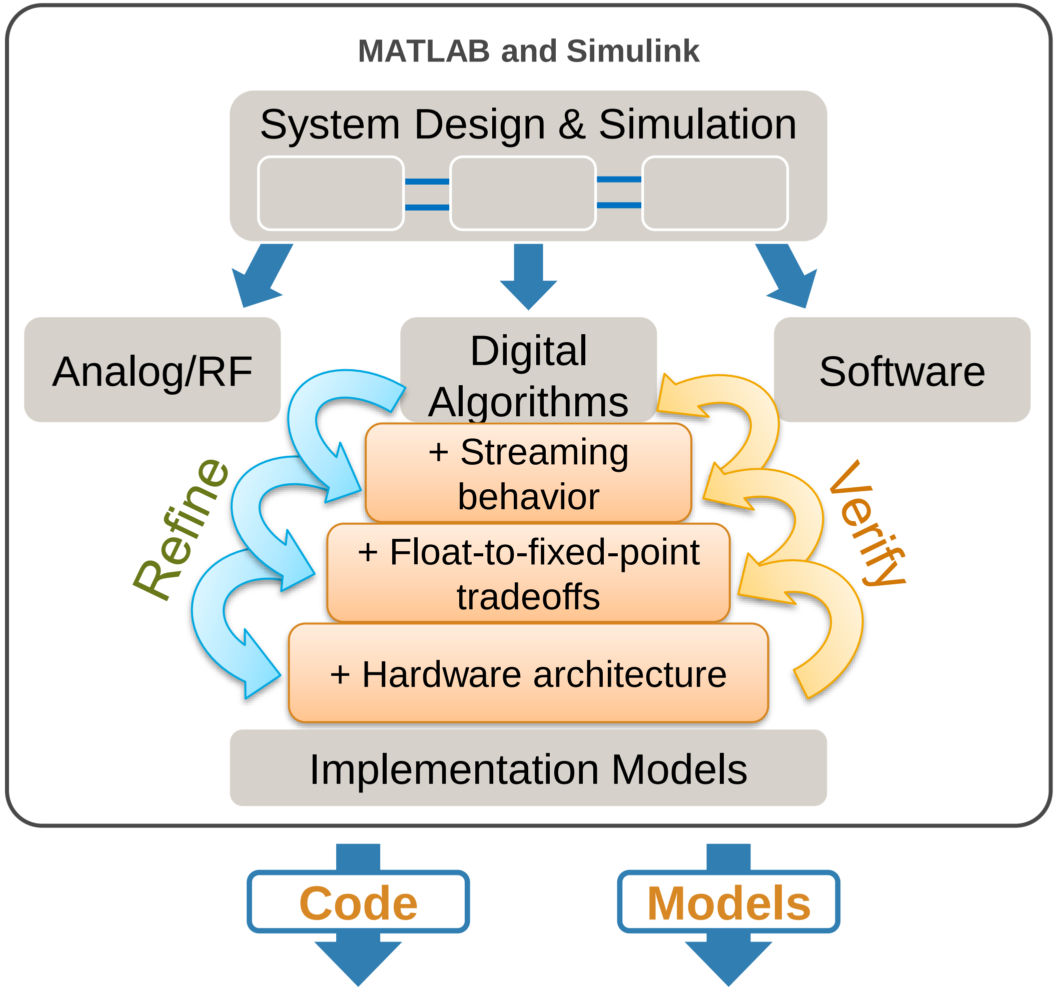
Checker:

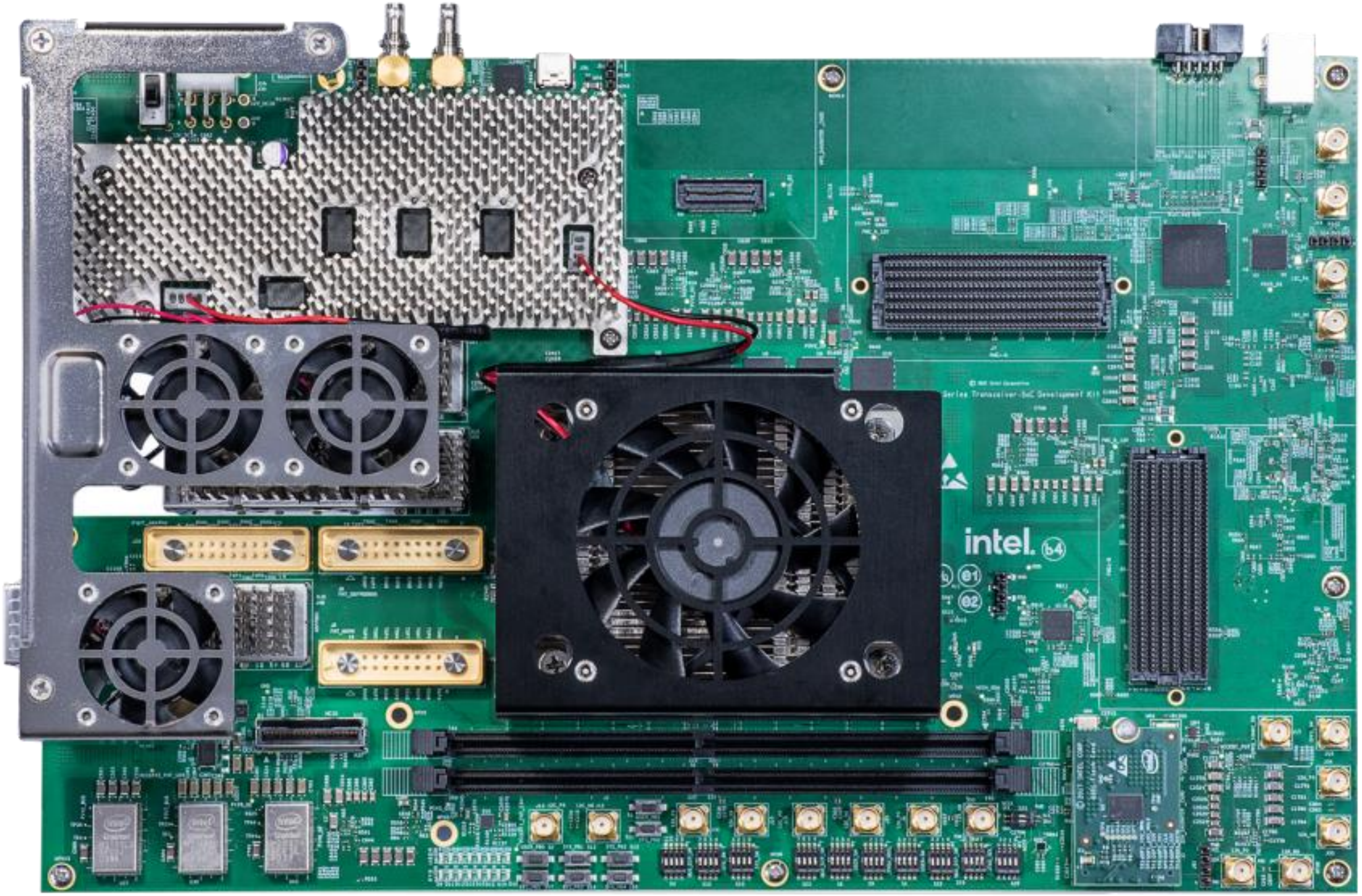
- <20 lines of MATLAB
- Easily adjusted

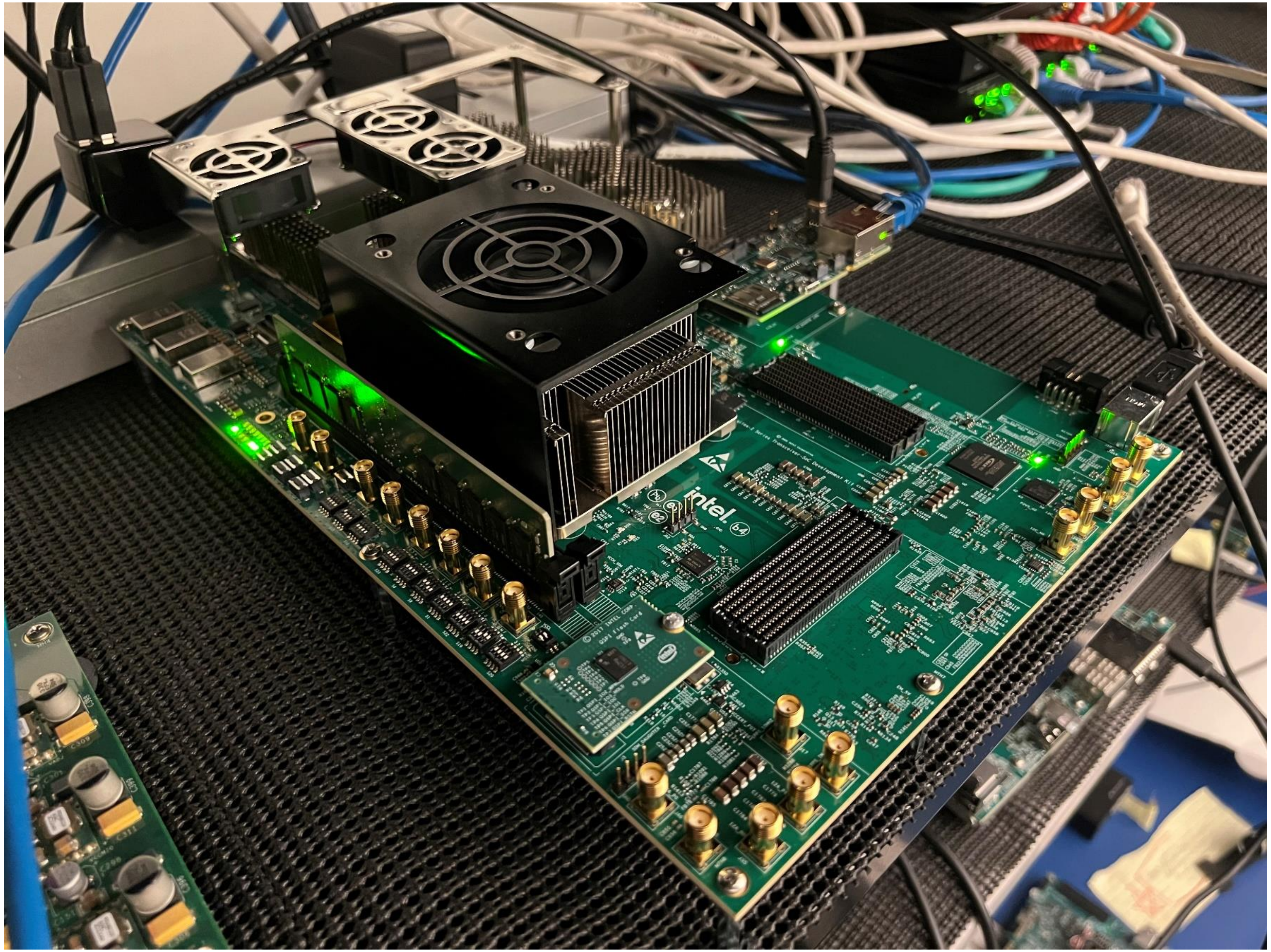
Get Started Collaborating to Speed Innovation in FPGA/ASIC Hardware

- All roles contribute to high-level design and exploration
- Eliminate costly system-level bugs early
- Generate error-free target-independent HDL
- Re-use algorithm development to speed verification and debug

> "Simulink helps system architects and hardware designers communicate. It is like a shared language that enables us to exchange knowledge, ideas, and designs."
>
> **Marcel van Bakel**
> **Philips Healthcare**




intel.


intel