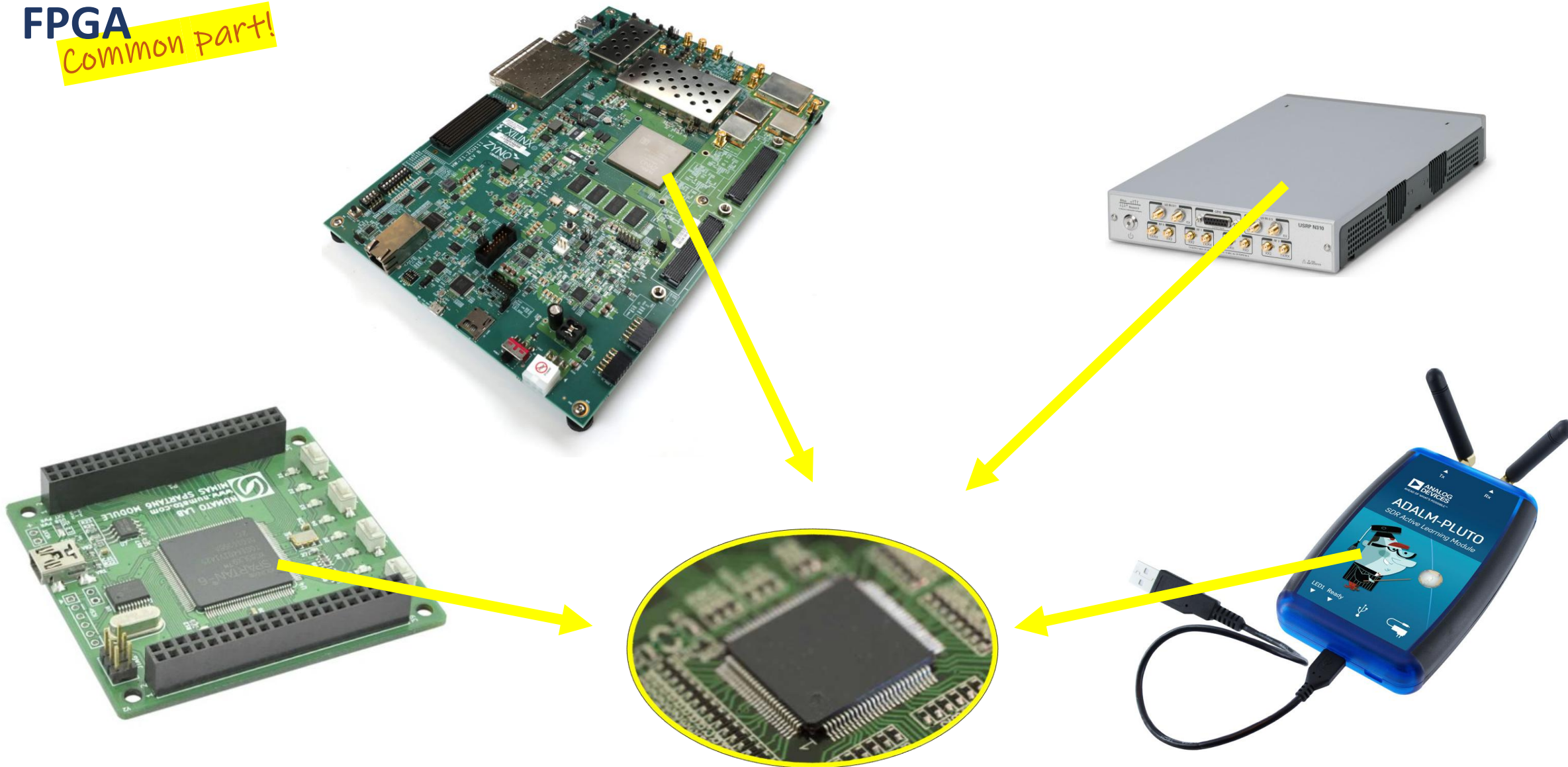


MATLAB EXPO

POLSKA 2024

FPGA
Common part!





Oprogramowanie
Naukowo-Techniczne
sp. z o.o.

MATLAB EXPO

Warszawa, 4.06.2024 r.

FPGA-based hardware support from MATLAB

Features and workflows

Mateusz Łabęcki, ONT

From MATLAB Algorithm to HDL Implementation

```

TxSignal = 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;

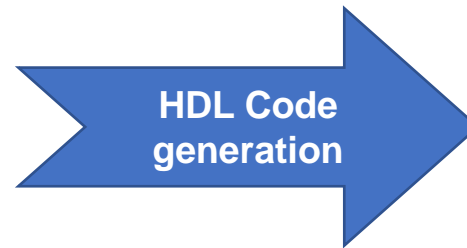
% Scale Rx signal to +/- one
scale1 = max([abs(real(RxSignal)); abs(imag(RxSignal))]);
RxSignal = RxSignal/scale1;

%% MATLAB golden reference

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

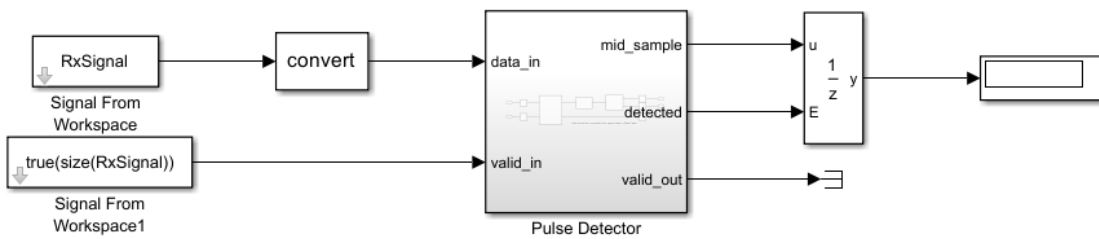
% Correlate Rx signal against matched filter
FilterOut = filter(CorrFilter,1,RxSignal);

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



```

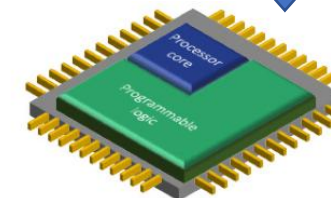
HDL Code
symmetric_fir.vhd (10) Search
1/10 <> X
124 ud2_process : PROCESS (clk, reset)
125 BEGIN
126 IF reset = '1' THEN
127   ud2_out1 <= to_signed(16#0000#, 16);
128 ELSIF clk'EVENT AND clk = '1' THEN
129   IF enb = '1' THEN
130     ud2_out1 <= ud1_out1;
131   END IF;
132 END IF;
133 END PROCESS ud2_process;
134
135
136 ud3_process : PROCESS (clk, reset)
137 BEGIN
138 IF reset = '1' THEN
139   ud3_out1 <= to_signed(16#0000#, 16);
140 ELSIF clk'EVENT AND clk = '1' THEN
141   IF enb = '1' THEN
142     ud3_out1 <= ud2_out1;
143   END IF;
144 END IF;
145 END PROCESS ud3_process;
146
  
```



© 2019-2022 The MathWorks, Inc.

Synthesizable RTL Code

*Synthesis
& Implementation*



Standalone FPGA



Code Generation with HDL Workflow Advisor

Workflow Advisor



Workflow
Advisor
ASSISTANCE

HDL Workflow Advisor - test_model_filter_design/Subsystem

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**
 - Run This Task
 - Run to Selected Task
 - Reset This Task
 - What's This?
 - 2. Prepare Model
 - 2.1. Check Model
 - 3. HDL Code Generation
 - 3.1. Set HDL Code Generation Options
 - ^3.2. Generate RTL Code and IP Core
 - 4. Embedded System Integration

1.4. Set Target Frequency

Analysis

Set Target Frequency

Input Parameters

Target Frequency (MHz): [0]

Default (MHz): [0] [Restore Default]

Frequency Range (MHz): [none]

[Run This Task]

Result: [Not Run]

To run this task, all prior tasks must have a result of Passed or Warning.

[Help] [Apply]

- HDL Workflow Advisor
 - 1. Set Target
 - ^1.1. Set Target Device and Synthesis Tool**

Result: **Passed**

Passed Set Target Device and Synthesis Tool.

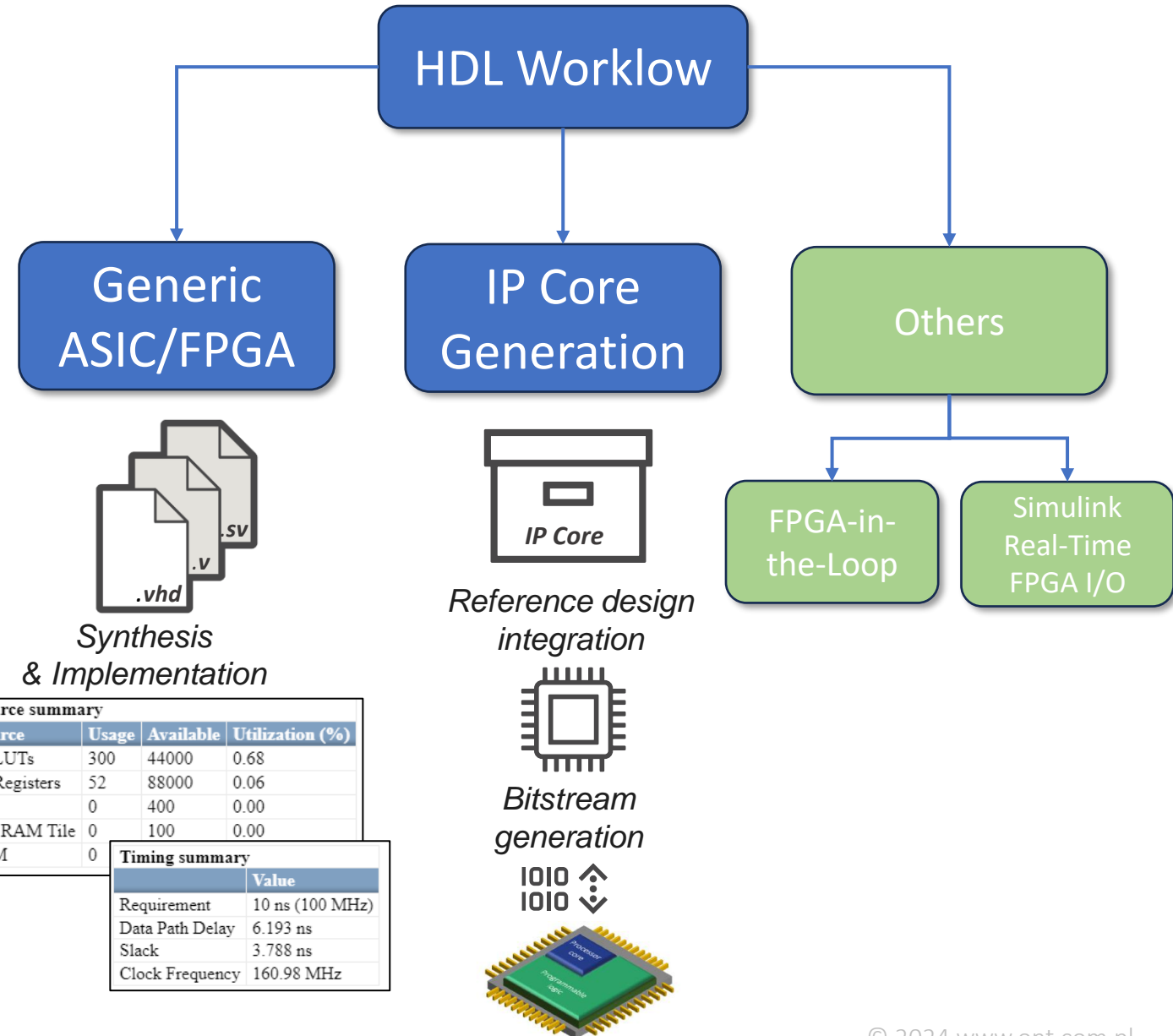
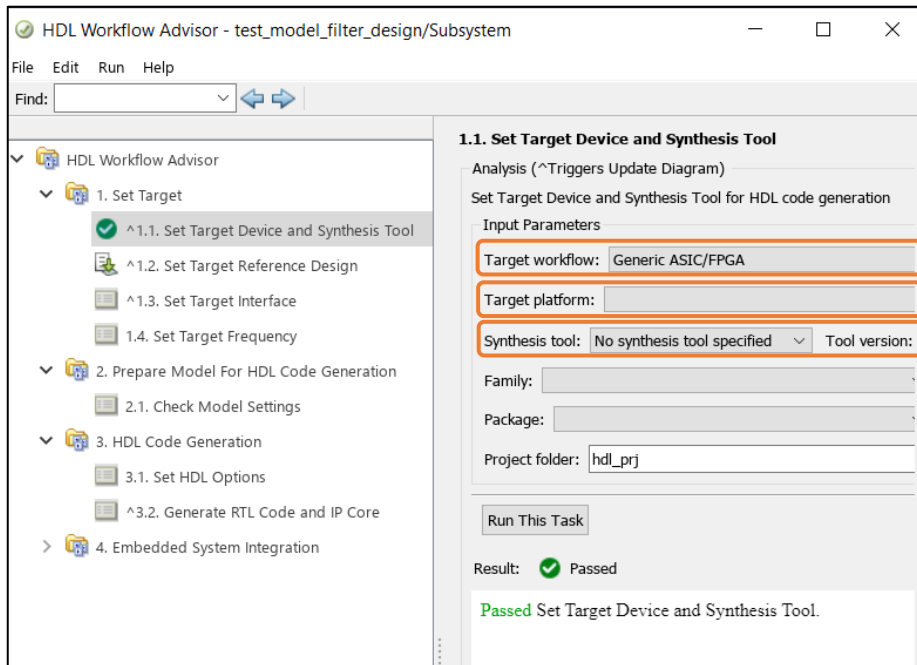
- HDL Workflow Advisor
 - 1. Set Target
 - ^1.1. Set Target Device and Synthesis Tool**

Result: **Failed**

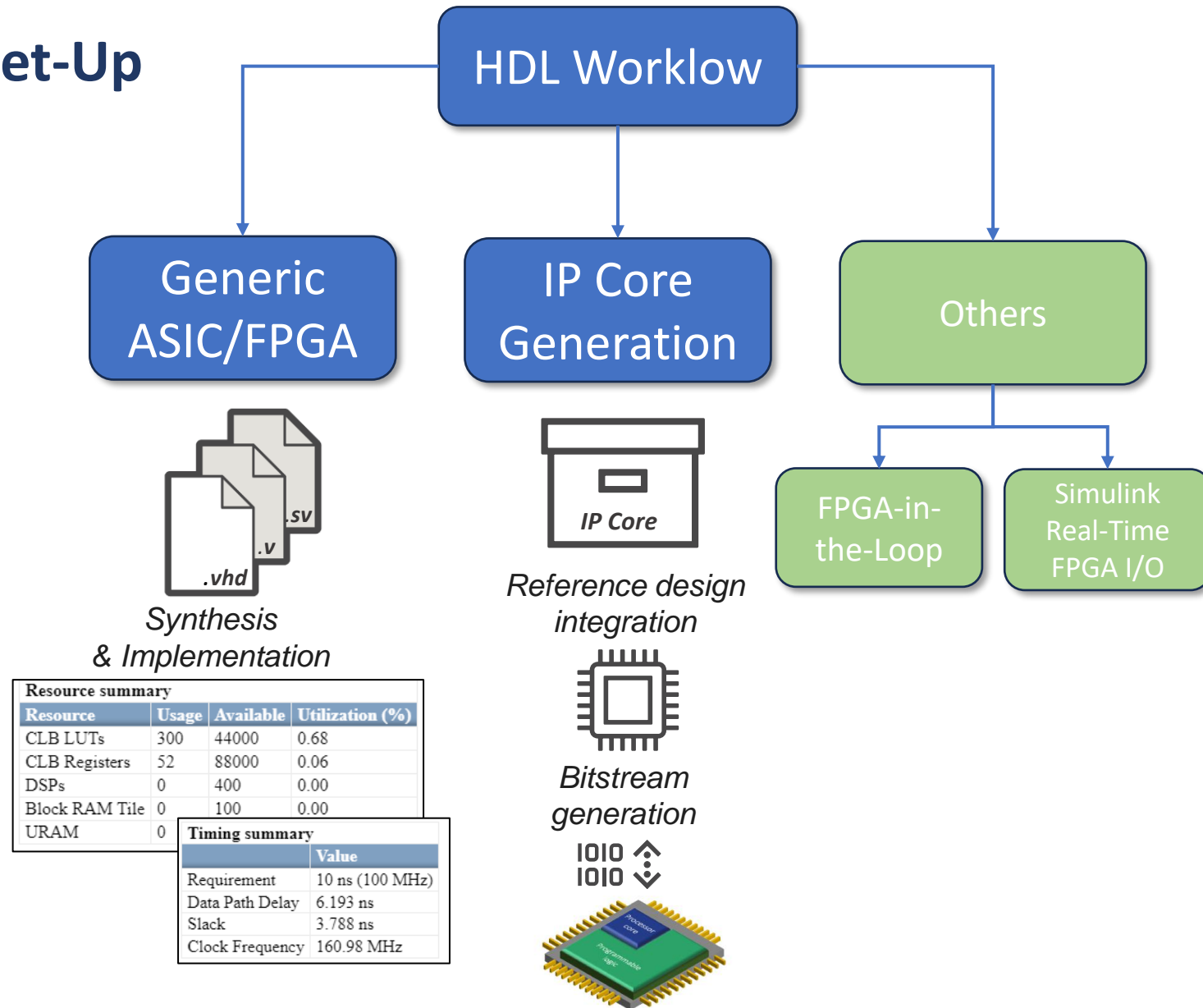
Failed IP Core Generation workflow requires the starting subsystem "test_model_filter_design/Subsystem" atomic subsystem. Please click the following link to [turn on "Treat as atomic unit"](#) on subsystem "test_model_filter_design/Subsystem".

HDL Workflow Set-Up

- Settings that affect Advisor's steps:
 - Target workflow
 - Target platform
 - Synthesis tool



HDL Workflow Set-Up



System-on-Chip



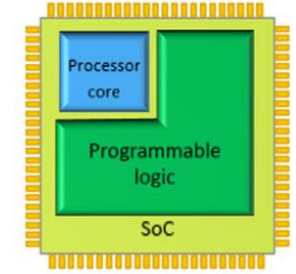
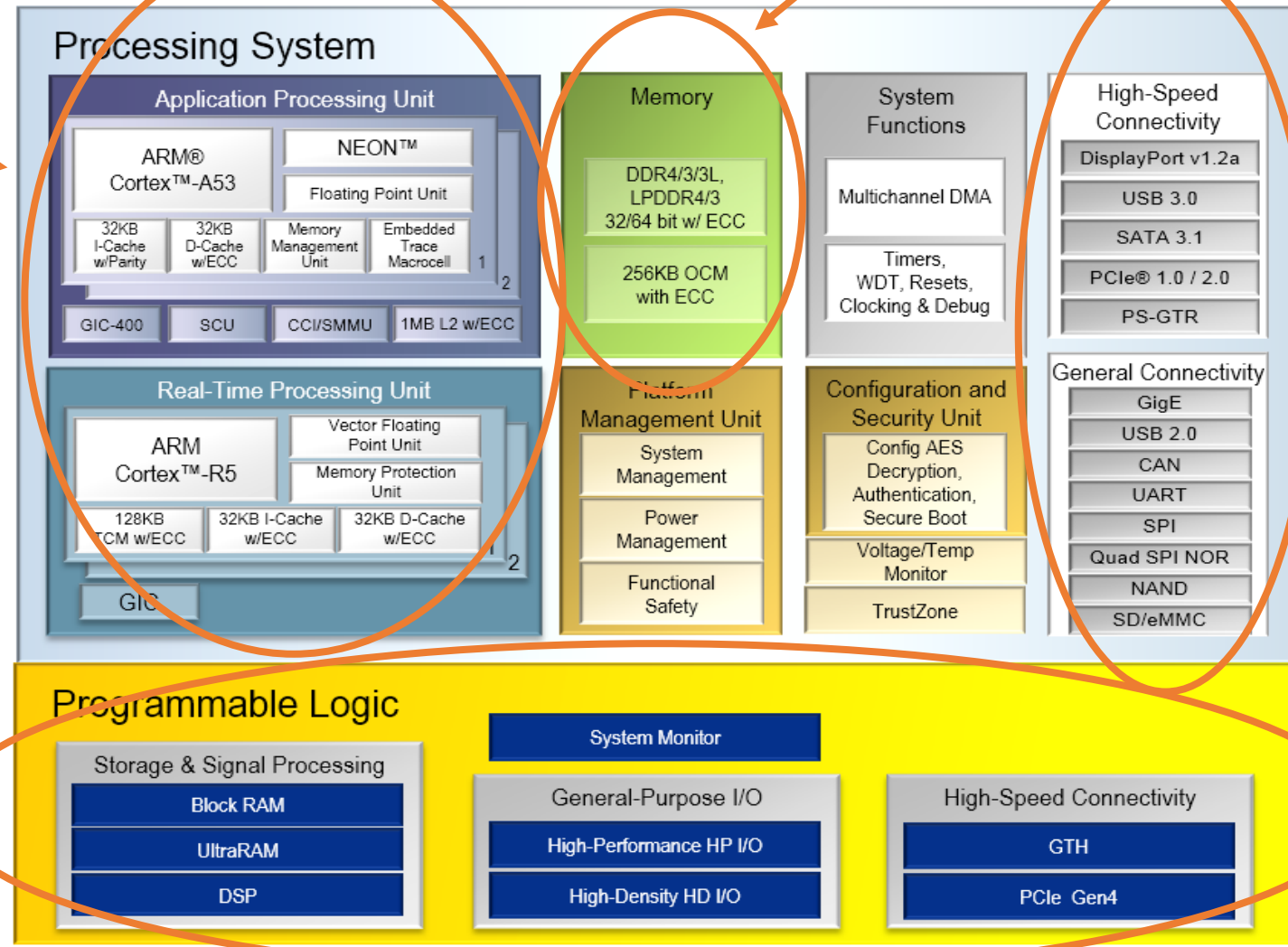
Targeting SoC

Multiple Processors

Memory Interfaces

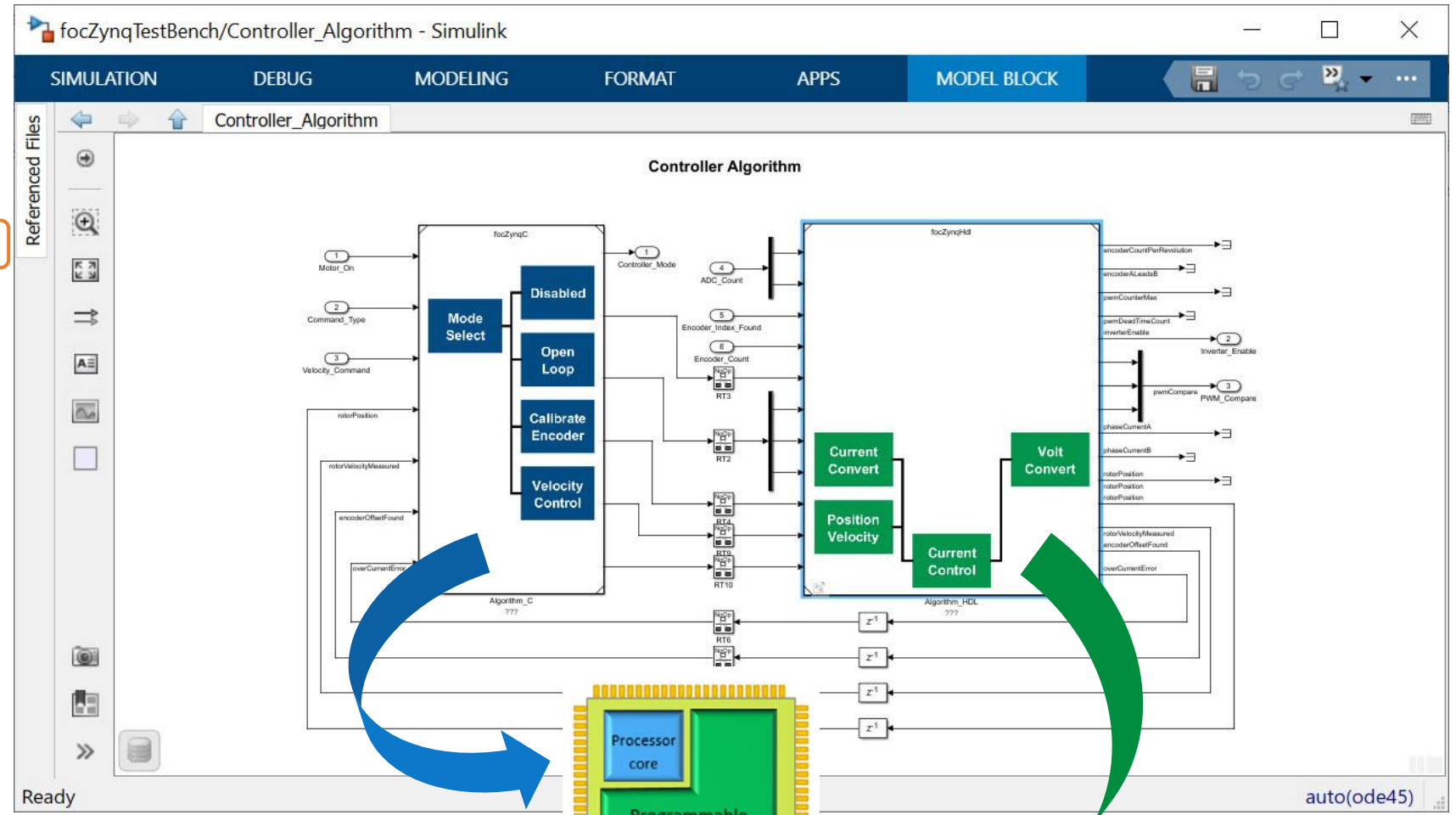
Programmable FPGA Logic

Communication Interfaces



Hardware-Software Partition

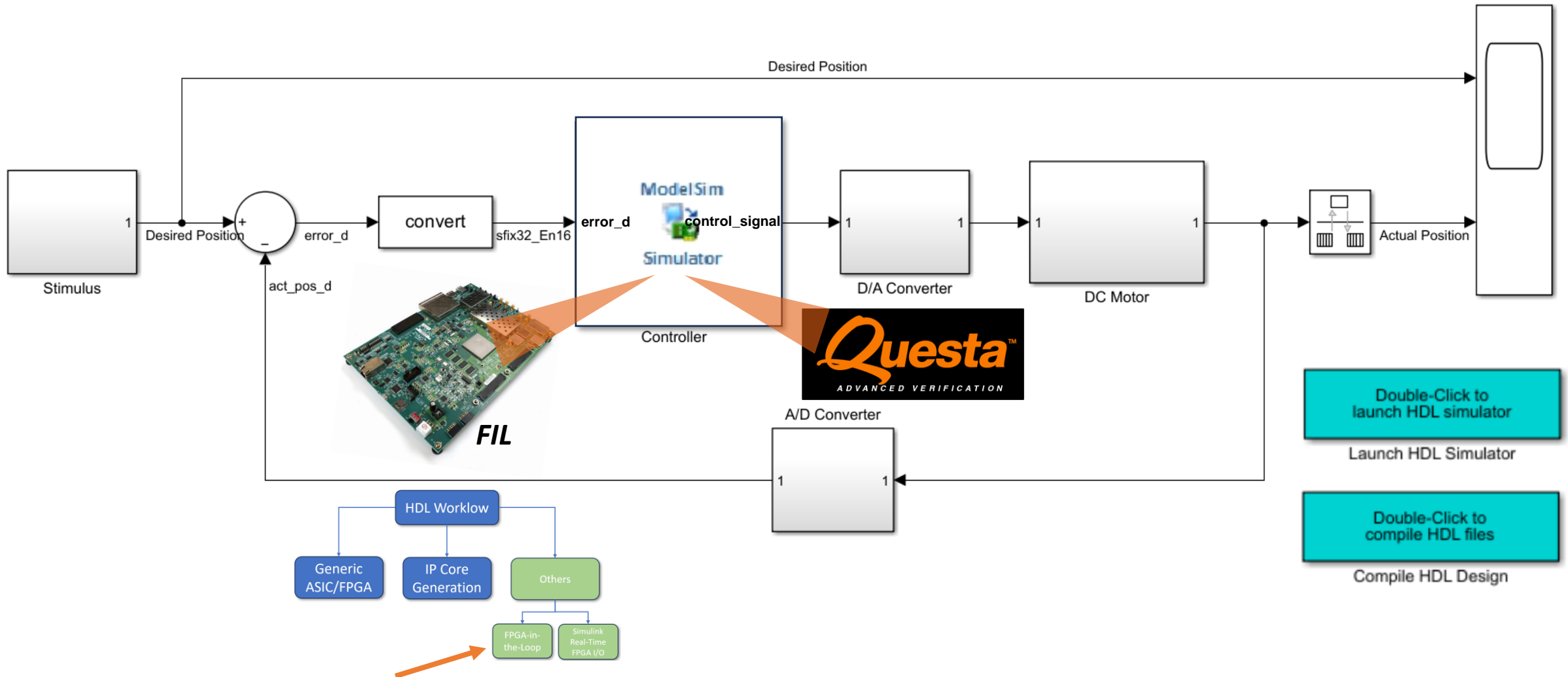
- 3. HDL Code Generation
 - 3.1. Set HDL Options
 - 3.2. Generate RTL Code and IP Core
- 4. Embedded System Integration
 - 4.1. Create Project
 - 4.2. Generate Software Interface
 - 4.3. Build FPGA Bitstream
 - 4.4. Program Target Device



Embedded Coder

HDL Coder

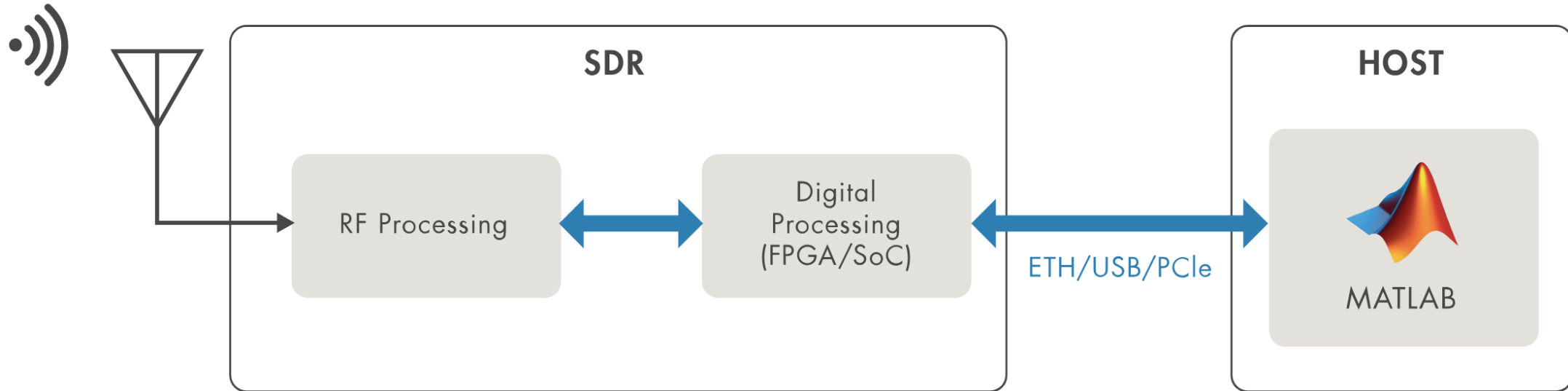
Next steps in MATLAB/Simulink – HDL Verification



Software-Defined Radio



Support from Communications Toolbox



ADALM-PLUTO Radio Support from Communications Toolbox

Prototype and Design of Software-Defined Radio (SDR) Systems using ADALM-PLUTO with MATLAB and Simulink

USRP E310 Support from Communications Toolbox

Prototype and Design of Software-Defined Radio (SDR) Systems using USRP E310 with MATLAB and Simulink

USRP Support from Communications Toolbox

Design and prototype software-defined radio (SDR) systems using USRP with MATLAB and Simulink

RTL-SDR Support from Communications Toolbox

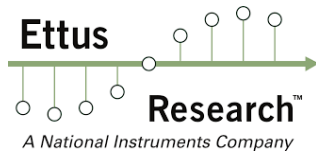
Design and prototype software-defined radio (SDR) systems using RTL-SDR with MATLAB and Simulink

Pluto Receiver	Receive data from Analog Devices ADALM-PLUTO radio
Pluto Transmitter	Transmit data to Analog Devices ADALM-PLUTO radio
RTL-SDR Receiver	Receive data from RTL-SDR device
E3xx Receiver	Receive data from USRP E3xx radio hardware (Since R2019b)
E3xx Transmitter	Send data to USRP E3xx radio hardware (Since R2019b)
SDRu Receiver	Receive data from USRP device
SDRu Transmitter	Send data to USRP device

Support for high-end USRPs from Wireless Testbench

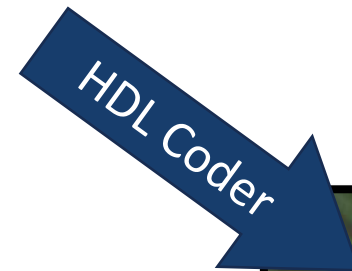
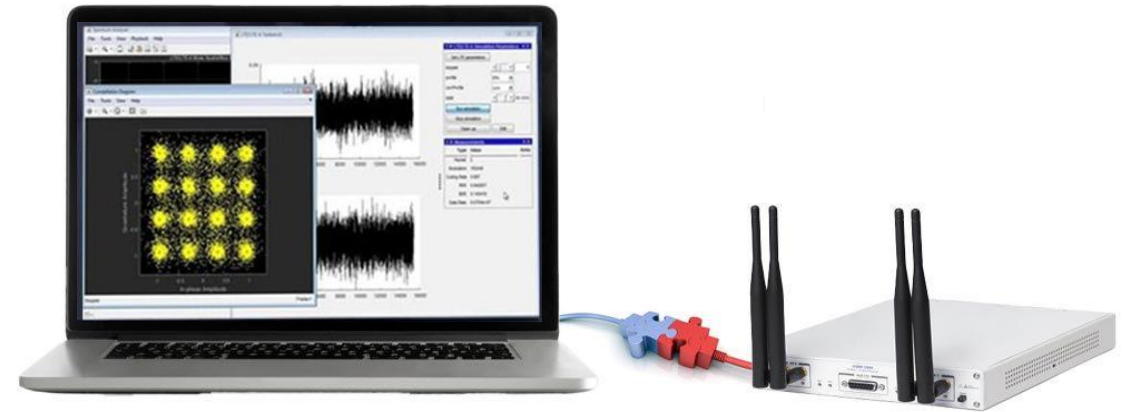
- Supported USRP Radios:

- X4xx
- X3xx
- N3xx



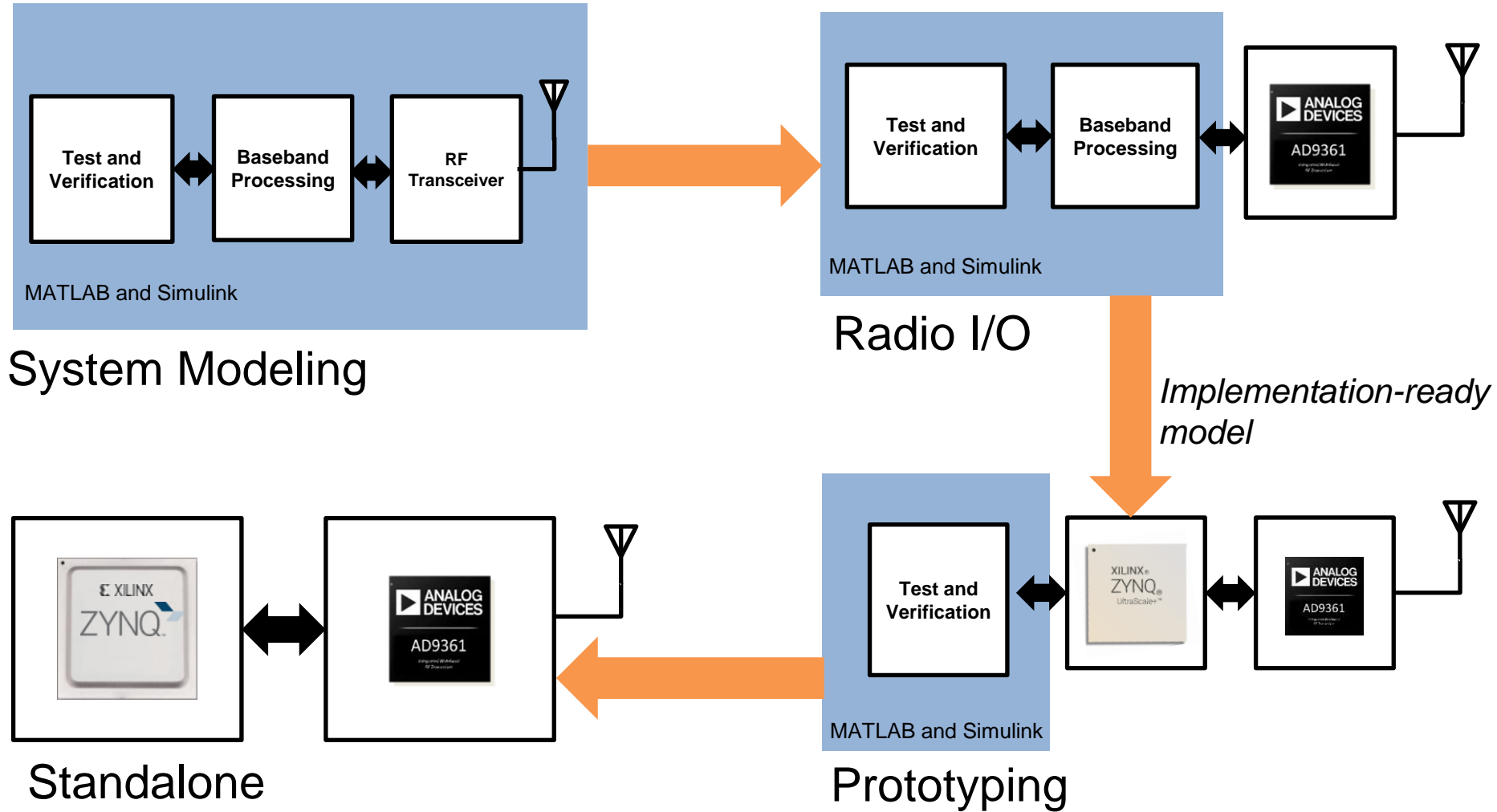
- Common Applications:

- Transmit and capture standard-based and custom wideband signals
- Intelligent signal detection using preamble and energy level
- Run prebuilt and custom wireless applications on the FPGA/SoC on USRP hardware



USRP FPGA Targeting

SDR Design Workflows

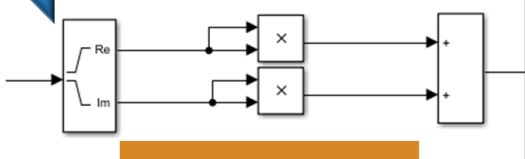


Further Exploration - Optimization

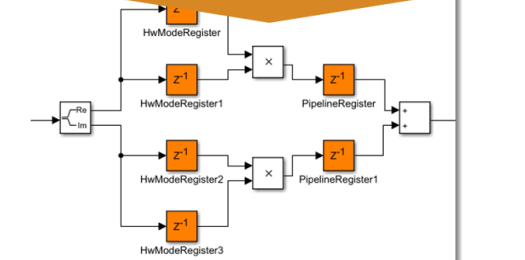
Automation ←

Control →

HW Arch Design



Adaptive Pipelining
Insert and balance



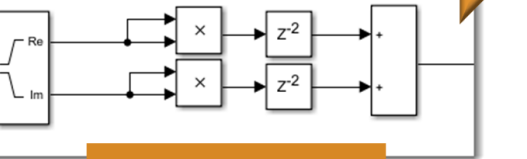
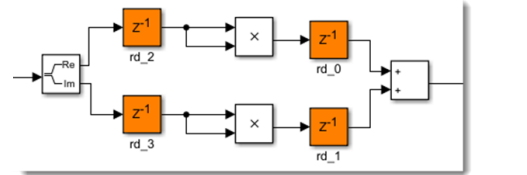
Resource summary		Timing summary	
Resource	Usage	Requirement	Value (ns)
CLB LUTs	0	Requirement	5
CLB Registers	0	Data Path Delay	2.049
DSPs	2	Slack	
Block RAM Tile	0		

Target Frequency

Generated Model

Synthesis & Implementation


Distributed Pipelining
Balance only

Resource summary		Timing summary	
Resource	Usage	Requirement	Value (ns)
CLB LUTs	0	Requirement	5
CLB Registers	0	Data Path Delay	2.049
DSPs	2	Slack	
Block RAM Tile	0		

Resource Sharing
Factor = 4

Inserts oversampled time-multiplexing logic



Block Parameters: FIR Filter

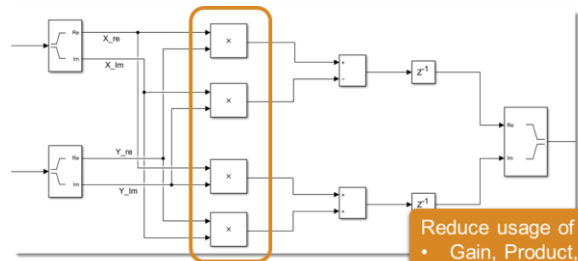
Biquad Filter (mask) (link)

Implement a general IIR filter using biquad structures. Biquad implementations of general IIR filters are often preferred due to their desirable numeric properties.

Coefficient source: Dialog parameters

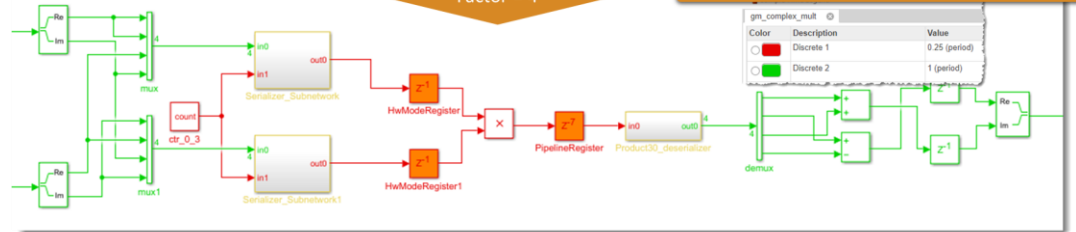
Main	Data Types
Filter structure:	Direct form II transposed
SOS Matrix (MxM):	Direct form I
	Direct form I transposed
	Direct form II
	Direct form II transposed

OK Cancel Help Apply



Reduce usage of expensive resources

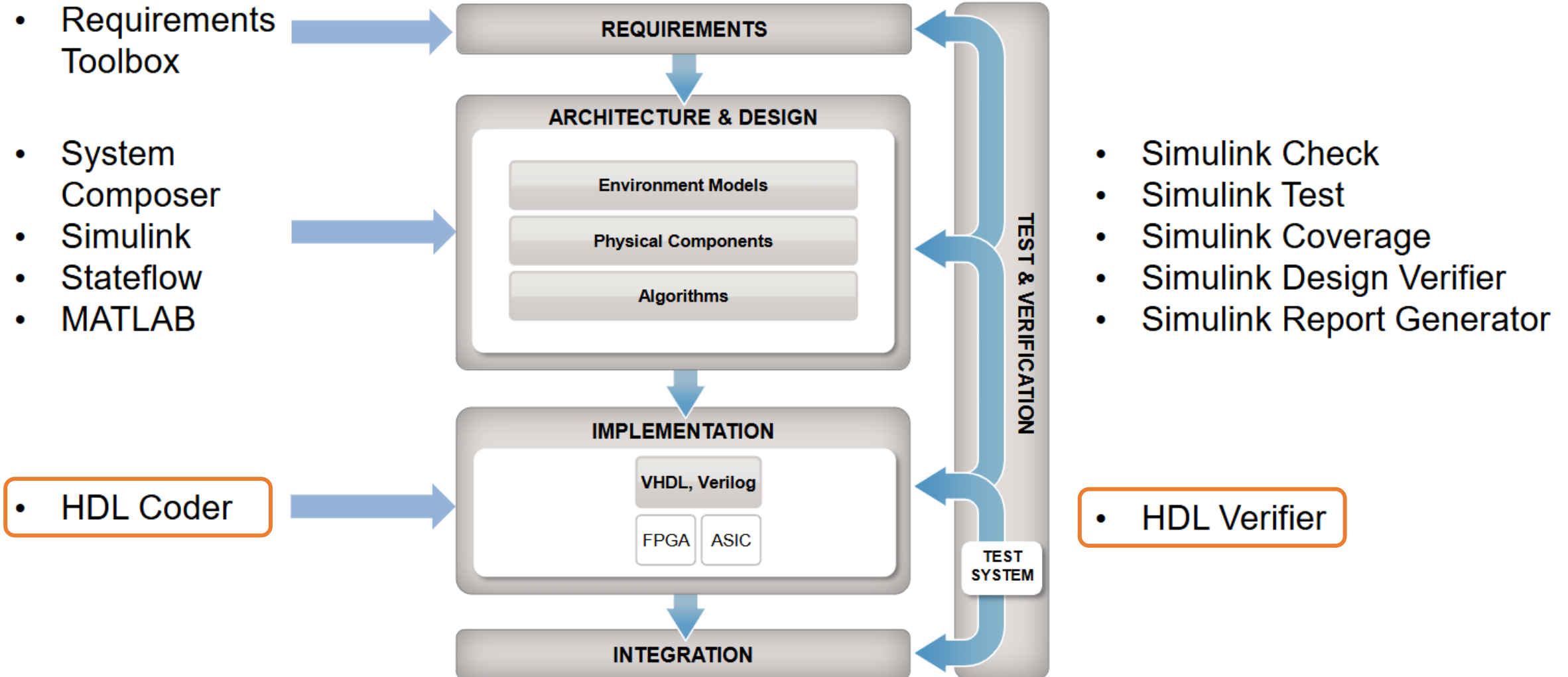
- Gain, Product, Multiply-Add



Color	Description	Value
●	Discrete 1	0.25 (period)
●	Discrete 2	1 (period)

- Timing optimization – pipelining
- Area optimization – resource sharing
- Both - Filter design optimization

Model-Based Design for FPGA



MATLAB EXPO

Warszawa | 4.06.2024

Mateusz Łabecki

Application Engineering Team Leader, ONT

mateusz.labecki@ont.com.pl

tel. +48 534 884 988

