MATLAB EXPO 2016

Ein Modell - viele Zielsysteme

Automatische Codegenerierung aus MATLAB und Simulink

Dr.-Ing. Daniel Weida

Industry trends

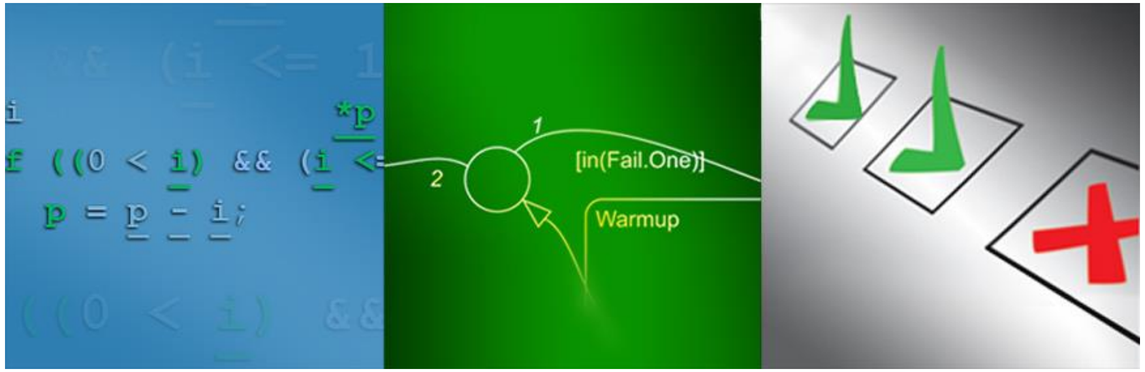
Code generation is expanding rapidly

- C
- C++

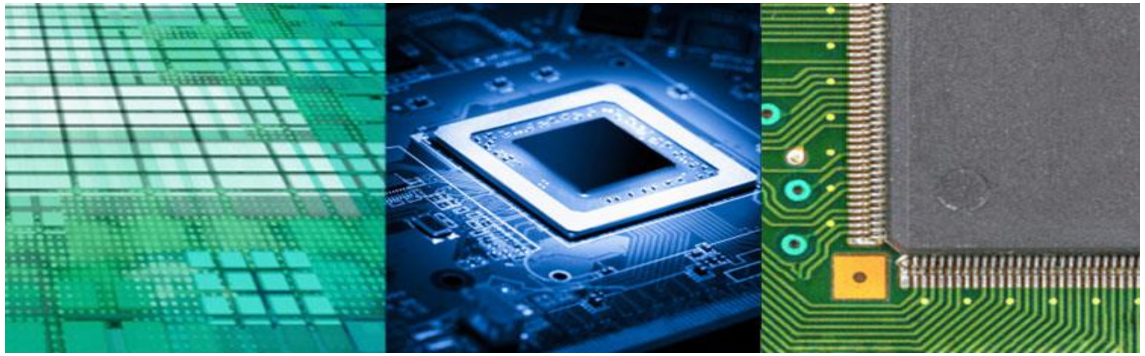
- VHDL
- Verilog

- Structured Text

Code generation offers many benefits



Hardware resources need optimization



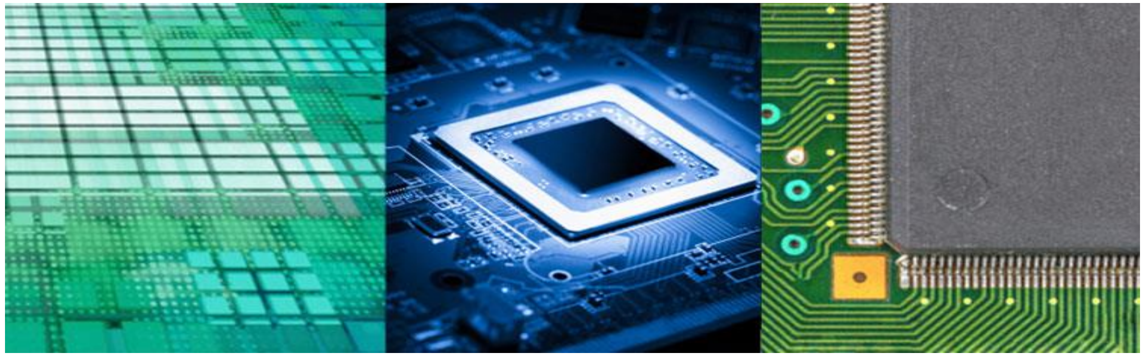
Agenda

Multi-target Production Code Generation

Hardware targets

Continuous Verification and Validation

Summary



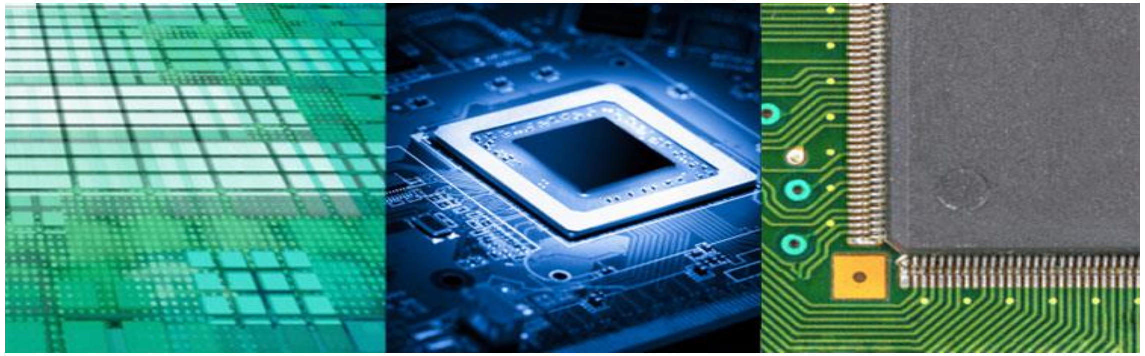
Agenda

Multi-target Production Code Generation

Hardware targets

Continuous Verification and Validation

Summary



Iveco Develops a Shift Range Inhibitor System for Mechanical 9- and 16-Speed Transmissions in Six Weeks

An Iveco heavy-duty vehicle.

Challenge

Develop and deliver an automotive transmission management system in six weeks

Solution

Use Model-Based Design to model, implement, test, and deploy the management system on a PLC

Results

- Development time cut by 40%
- Specification and implementation errors eliminated
- PLC design reused on a microprocessor

> "Our system engineers work directly with our software engineers on the Simulink model. This speeds development because there is no misinterpretation of requirements. When we're confident that the model is right, we save even more time by generating code from it, with no implementation errors."
>
> Demetrio Cortese
> Iveco

Multi-target challenges

Model-Based Design

- How do I size the motors?
- Can I get desired performance?
- Does my system work if component values change?

Multi-target Production Code Generation

- How do I size the processing hardware?
- Can I get desired execution speed?
- Does the system work if component cores change (e.g., PLC to DSP)?

> **“After implementing the PLC version with Simulink PLC Coder, we reused the model, with few modifications, and generated the microprocessor code using Embedded Coder. We switched from a structured text implementation to C, just by changing the code generation product we used.”**
>
> **Demetrio Cortese, Iveco**

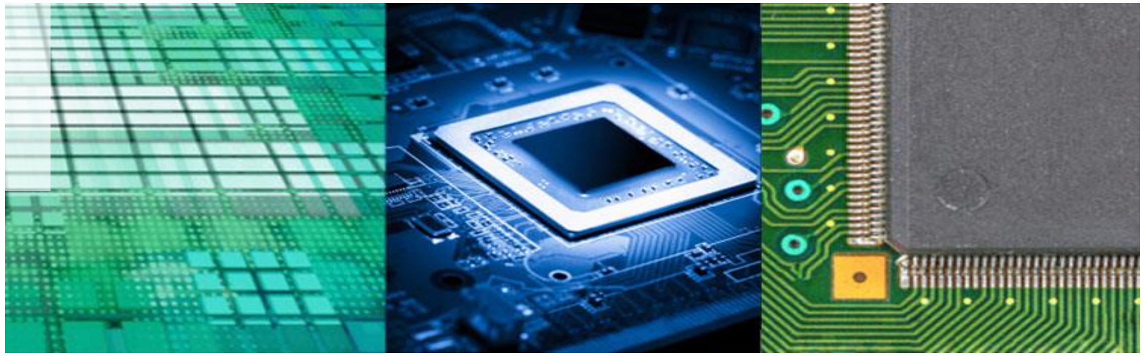
Agenda

Multi-target Production Code Generation

Hardware targets

Continuous Verification and Validation

Summary

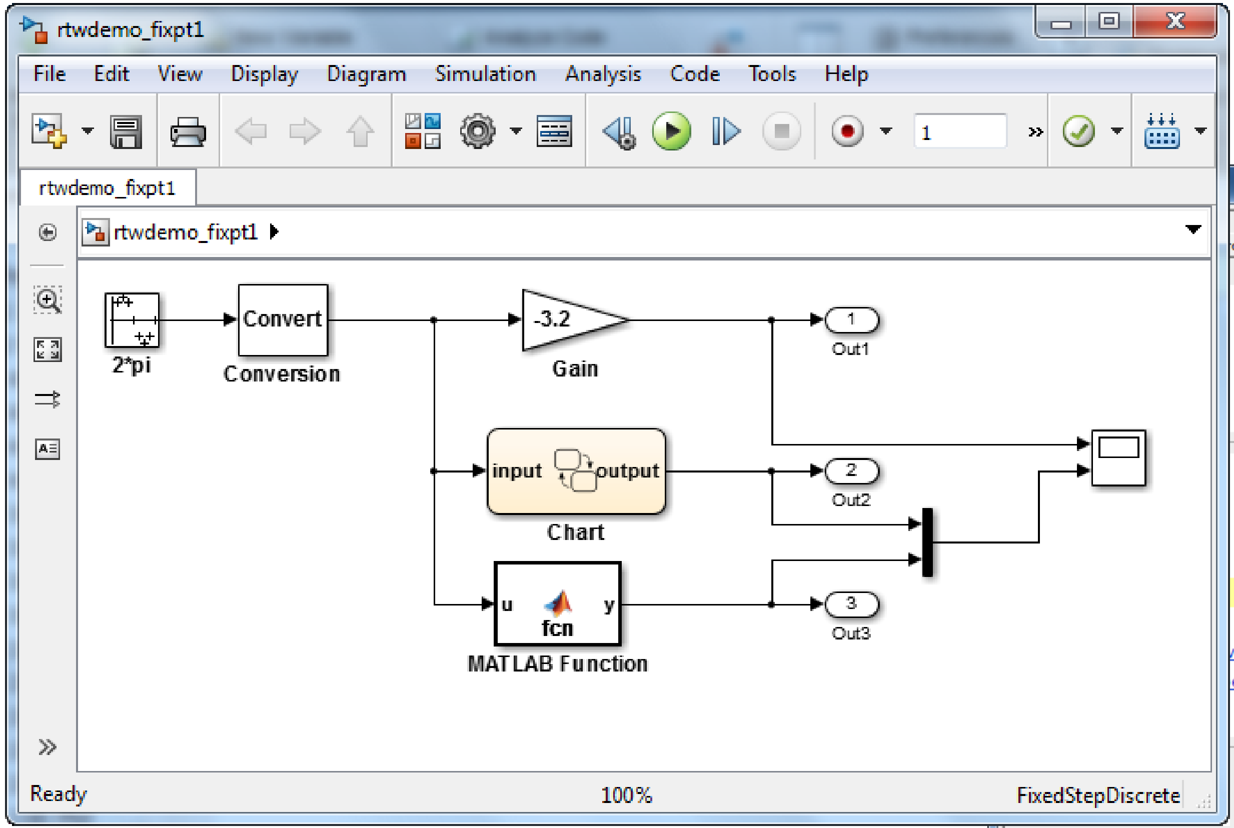


Code Generation: Five languages

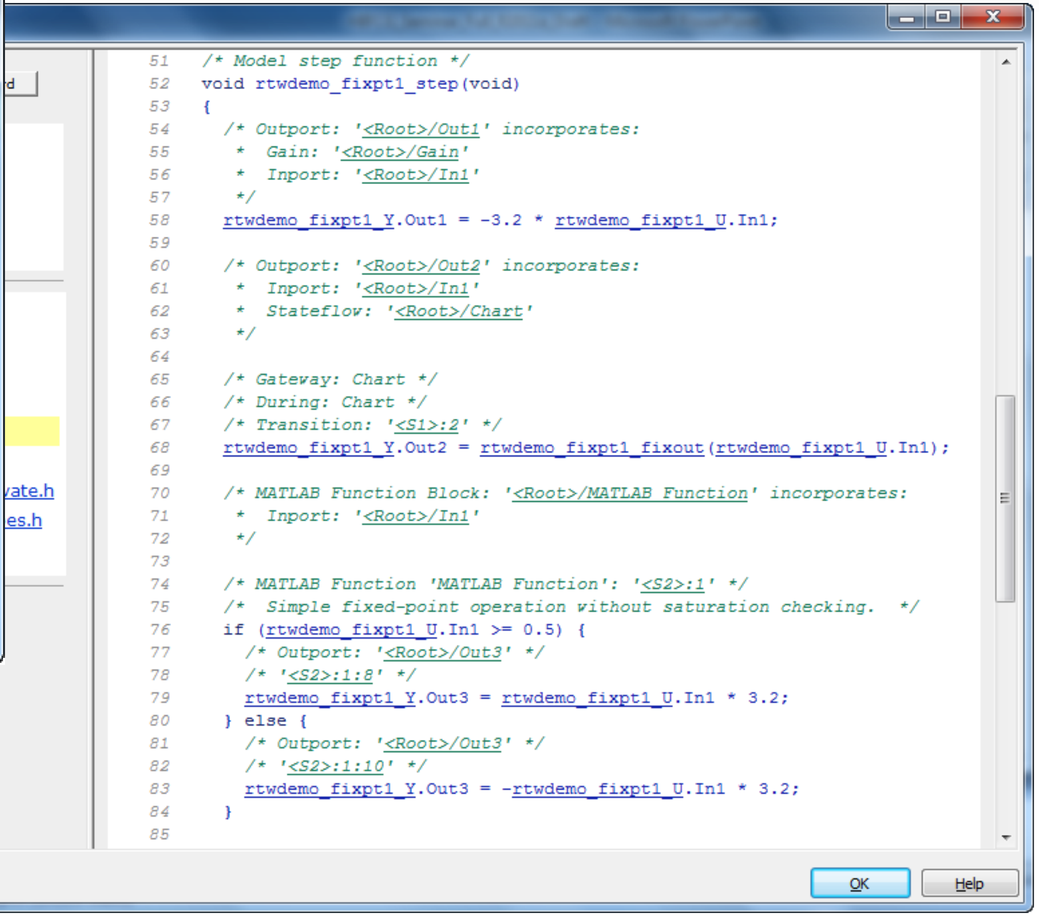
- C
- C++

- VHDL
- Verilog

- Structured Text



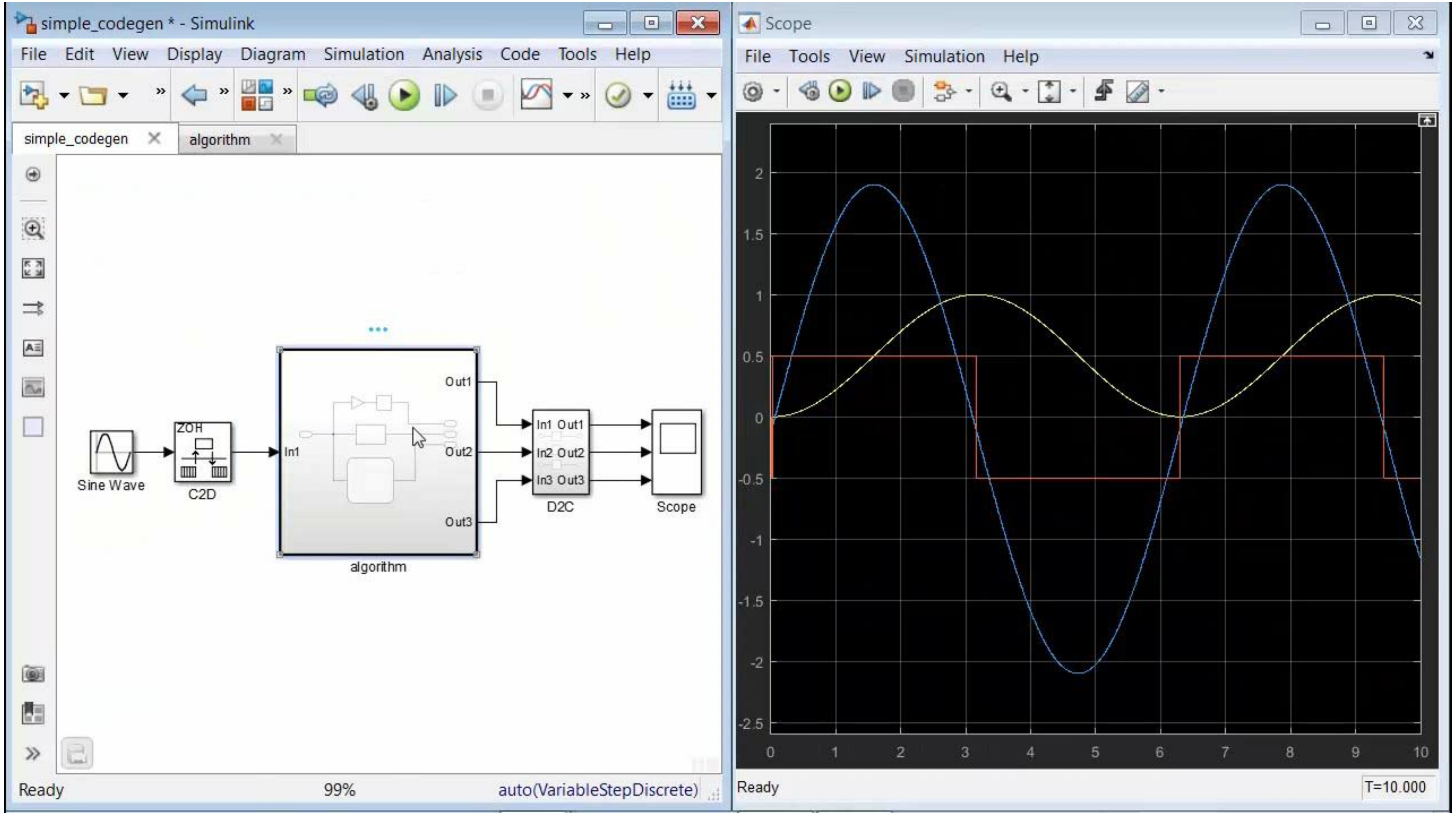
Model



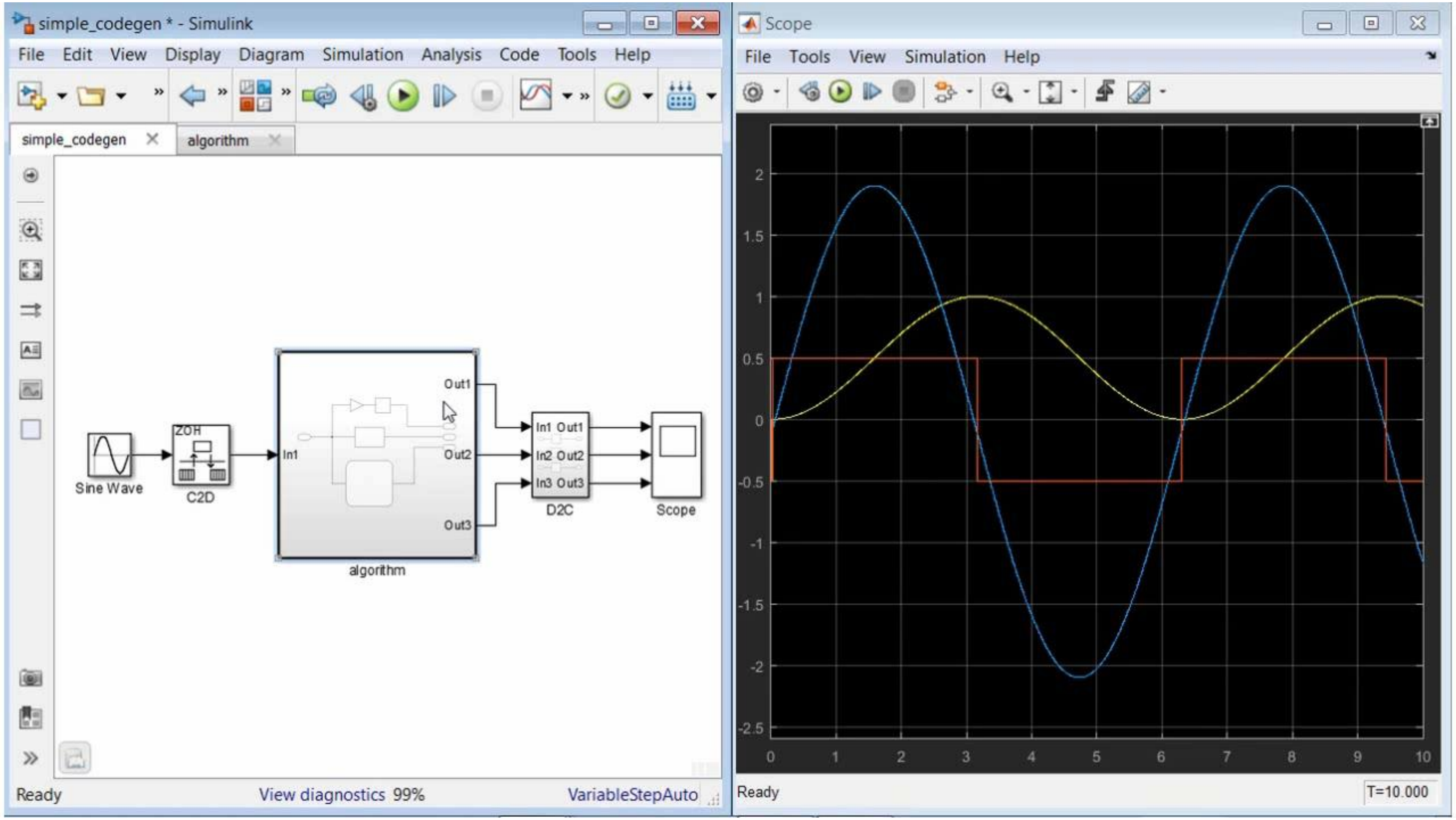
```
/* Model step function */
void rtwdemo_fixpt1_step(void)
{
  /* Outport: '<Root>/Out1' incorporates:
   *  Gain: '<Root>/Gain'
   *  Inport: '<Root>/In1'
   */
  rtwdemo_fixpt1_Y.Out1 = -3.2 * rtwdemo_fixpt1_U.In1;

  /* Outport: '<Root>/Out2' incorporates:
   *  Inport: '<Root>/In1'
   *  Stateflow: '<Root>/Chart'
   */

  /* Gateway: Chart */
  /* During: Chart */
  /* Transition: '<S1>:2' */
  rtwdemo_fixpt1_Y.Out2 = rtwdemo_fixpt1_fixout(rtwdemo_fixpt1_U.In1);

  /* MATLAB Function Block: '<Root>/MATLAB Function' incorporates:
   *  Inport: '<Root>/In1'
   */

  /* MATLAB Function 'MATLAB Function': '<S2>:1' */
  /*  Simple fixed-point operation without saturation checking.  */
  if (rtwdemo_fixpt1_U.In1 >= 0.5) {
    /* Outport: '<Root>/Out3' */
    /* '<S2>:1:8' */
    rtwdemo_fixpt1_Y.Out3 = rtwdemo_fixpt1_U.In1 * 3.2;
  } else {
    /* Outport: '<Root>/Out3' */
    /* '<S2>:1:10' */
    rtwdemo_fixpt1_Y.Out3 = -rtwdemo_fixpt1_U.In1 * 3.2;
  }

```

Code

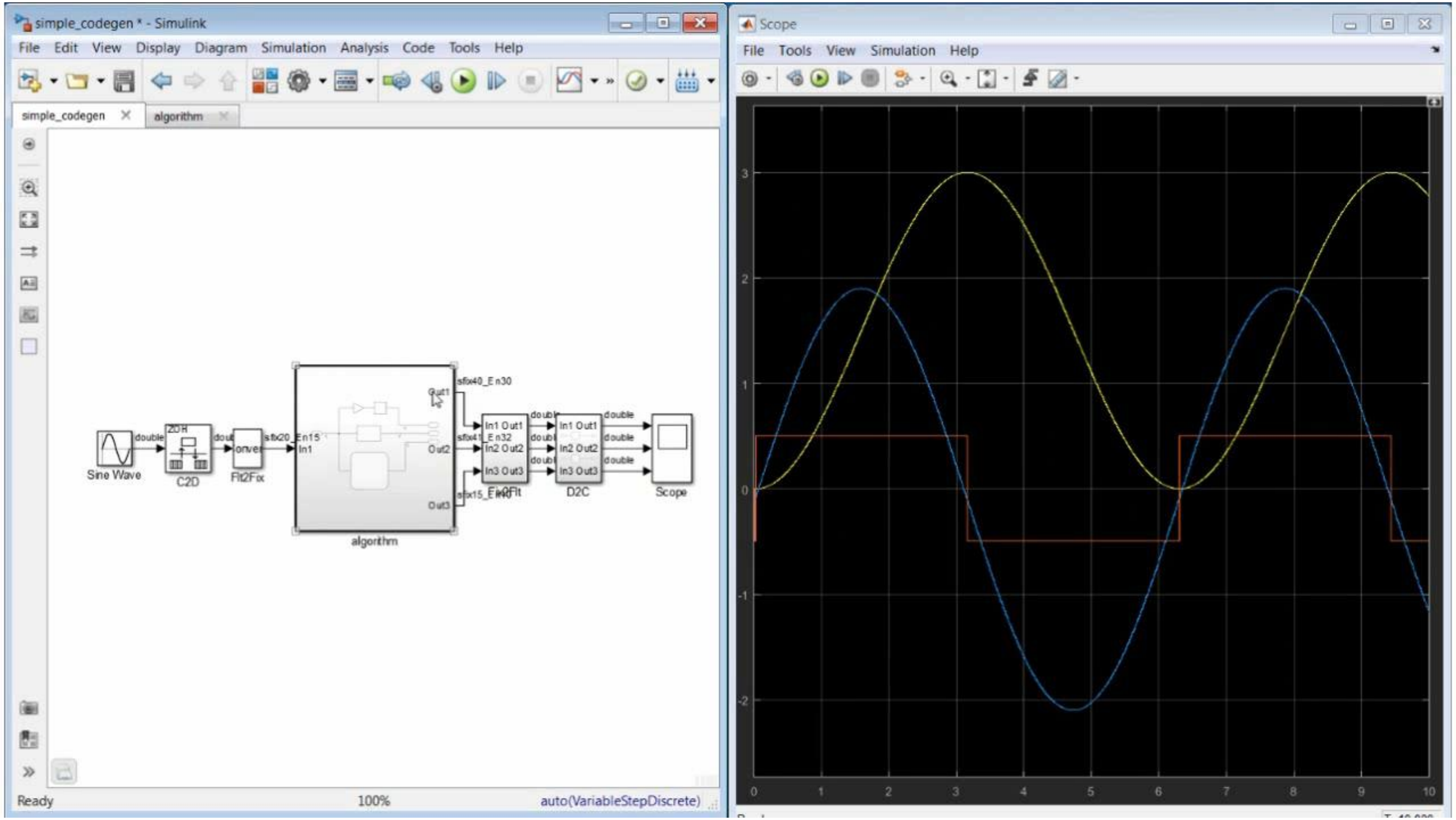
Structured Text



C/C++


simple_codegen * - Simulink
File Edit View Display Diagram Simulation Analysis Code Tools Help
simple_codegen
algorithm
Sine Wave
ZOH
C2D
In1
Out1
Out2
Out3
algorithm
In1 Out1
In2 Out2
In3 Out3
D2C
Scope
Ready
View diagnostics 99%
VariableStepAuto
Scope
File Tools View Simulation Help
2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
-2.5
0 1 2 3 4 5 6 7 8 9 10
Ready
T=10.000

VHDL/Verilog

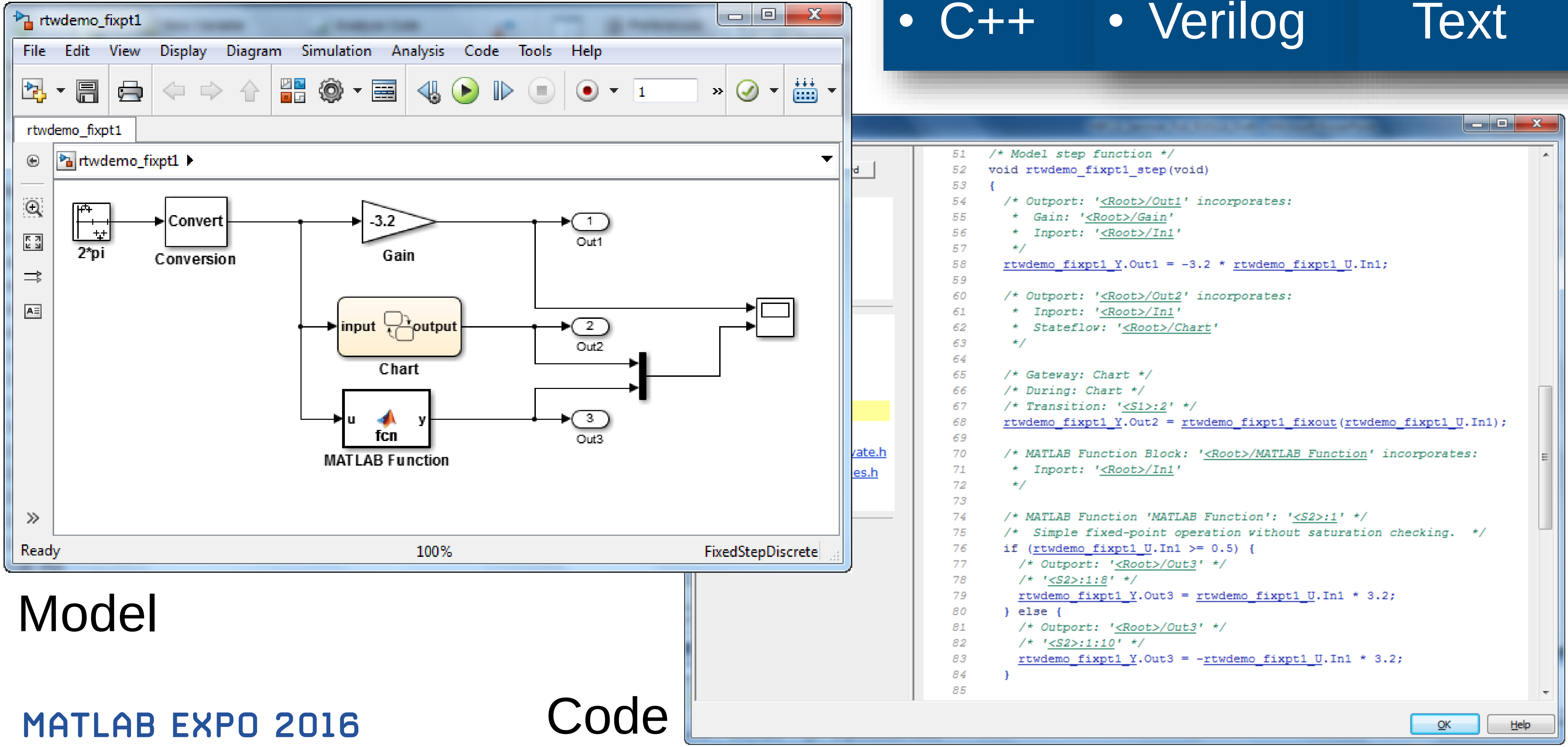


Code Generation: Five languages

- C
- C++

- VHDL
- Verilog

- Structured Text



```
/* Model step function */
void rtwdemo_fixpt1_step(void)
{
  /* Outport: '<Root>/Out1' incorporates:
   *  Gain: '<Root>/Gain'
   *  Inport: '<Root>/In1'
   */
  rtwdemo_fixpt1_Y.Out1 = -3.2 * rtwdemo_fixpt1_U.In1;

  /* Outport: '<Root>/Out2' incorporates:
   *  Inport: '<Root>/In1'
   *  Stateflow: '<Root>/Chart'
   */

  /* Gateway: Chart */
  /* During: Chart */
  /* Transition: '<S1>:2' */
  rtwdemo_fixpt1_Y.Out2 = rtwdemo_fixpt1_fixout(rtwdemo_fixpt1_U.In1);

  /* MATLAB Function Block: '<Root>/MATLAB Function' incorporates:
   *  Inport: '<Root>/In1'
   */

  /* MATLAB Function 'MATLAB Function': '<S2>:1' */
  /*  Simple fixed-point operation without saturation checking.  */
  if (rtwdemo_fixpt1_U.In1 >= 0.5) {
    /* Outport: '<Root>/Out3' */
    /* '<S2>:1:8' */
    rtwdemo_fixpt1_Y.Out3 = rtwdemo_fixpt1_U.In1 * 3.2;
  } else {
    /* Outport: '<Root>/Out3' */
    /* '<S2>:1:10' */
    rtwdemo_fixpt1_Y.Out3 = -rtwdemo_fixpt1_U.In1 * 3.2;
  }

```

Model

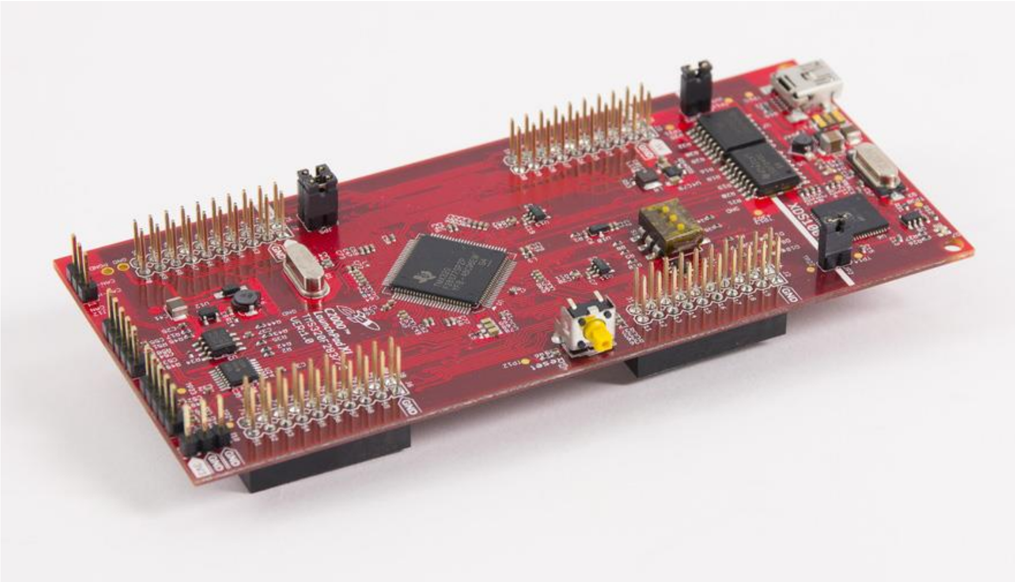
Code

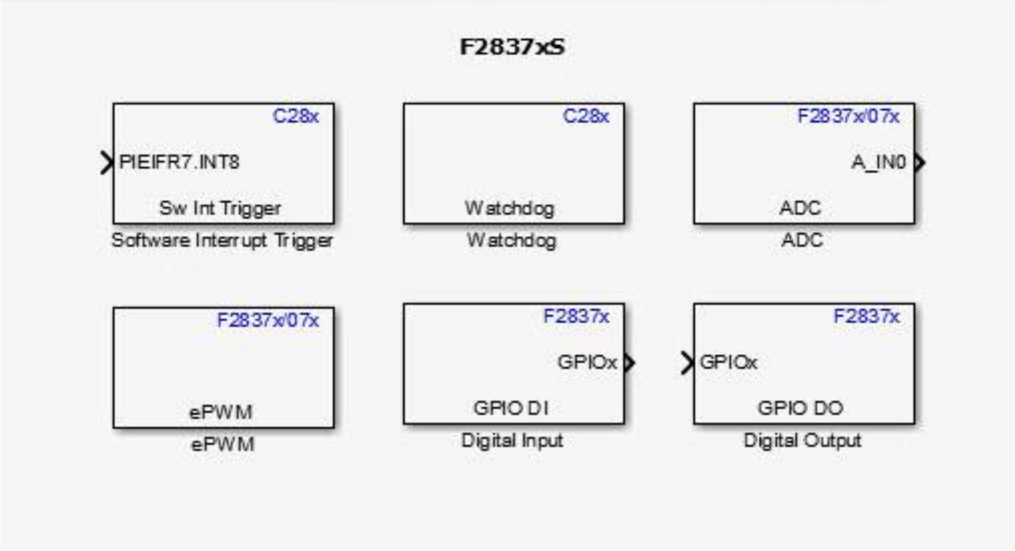
Hardware Support: Any device

- Any device with portable code for **algorithm code generation**

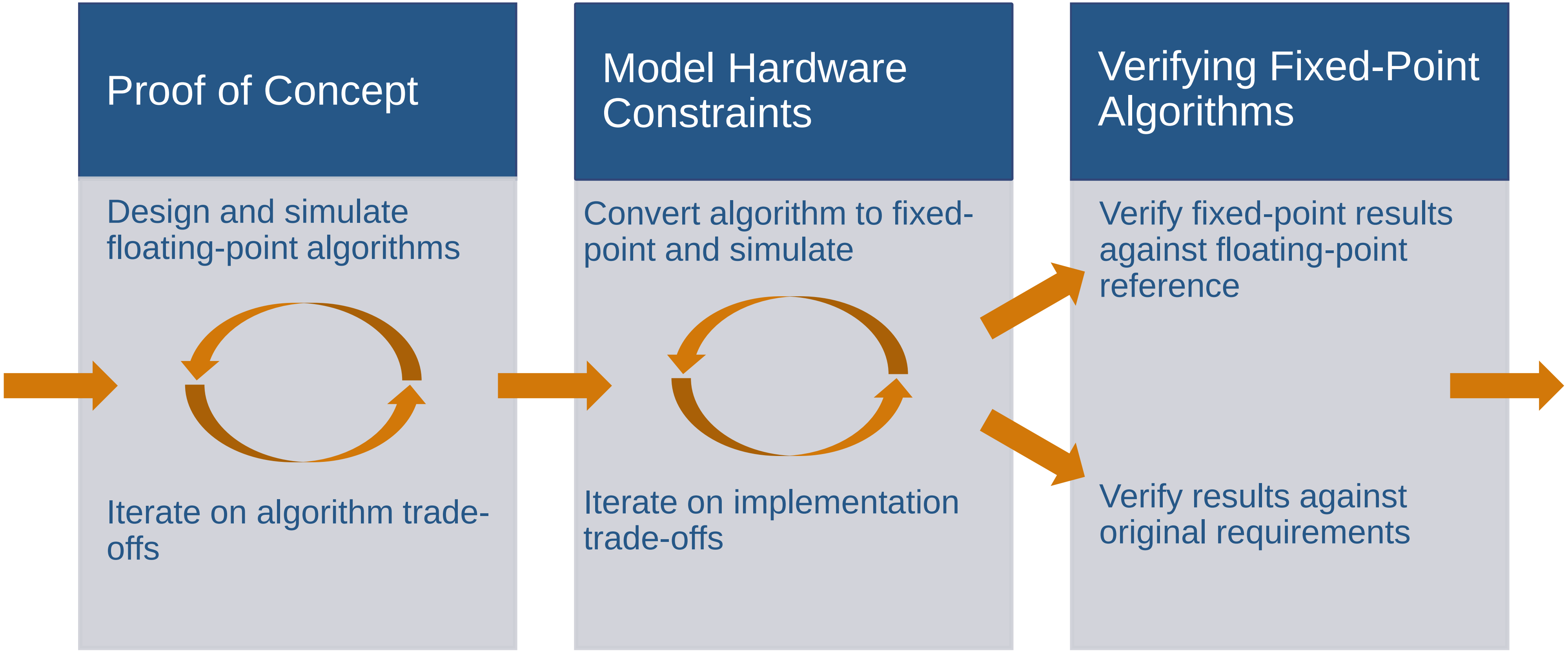
- Coder support packages offer device-specific **system executable generation**
  - ARM … C2000 … Zynq

- Hardware vendors offer their own support packages
  - Freescale, Infineon, Microchip, Renesas, TI, STMicroelectronics, …





Hardware Constraints



Proof of Concept

Design and simulate floating-point algorithms

Iterate on algorithm trade-offs

Model Hardware Constraints

Convert algorithm to fixed-point and simulate

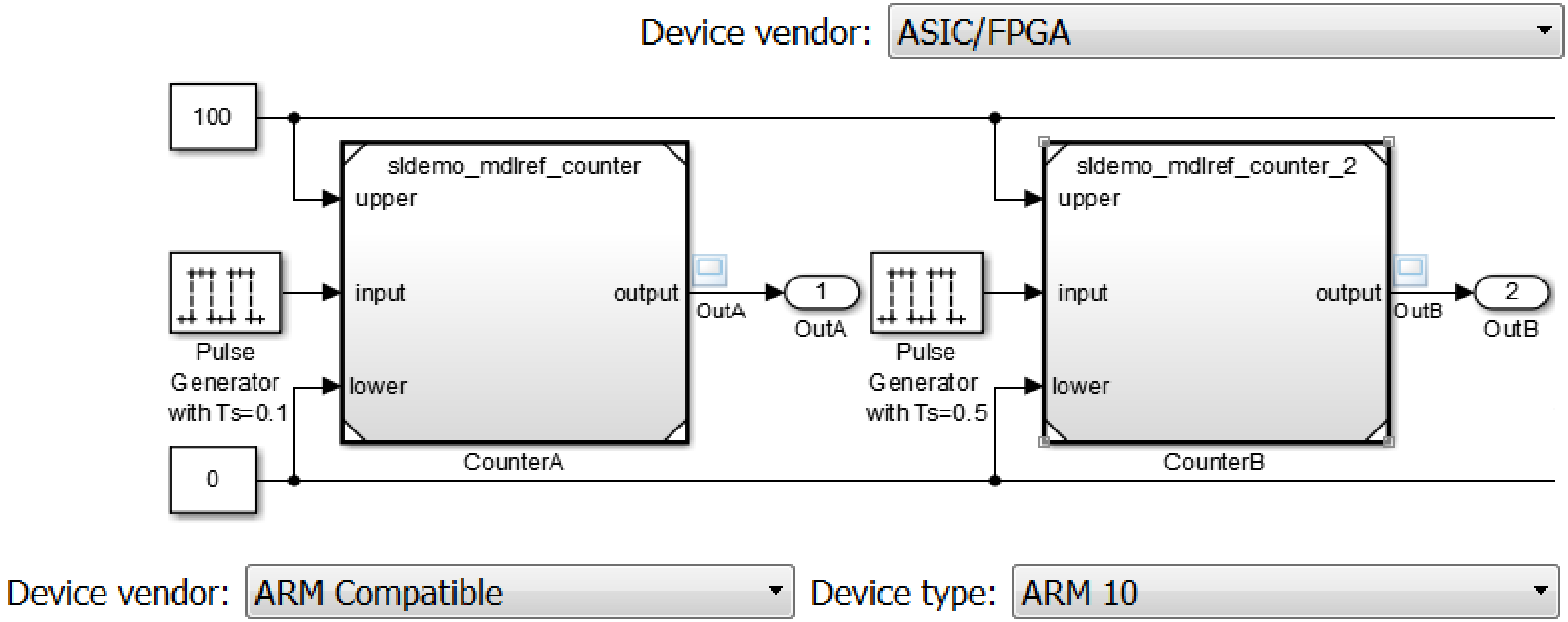
Iterate on implementation trade-offs

Verifying Fixed-Point Algorithms

Verify fixed-point results against floating-point reference

Verify results against original requirements

Simulation for Mixed Targets



Device vendor: ASIC/FPGA

Device vendor: ARM Compatible

Device type: ARM 10

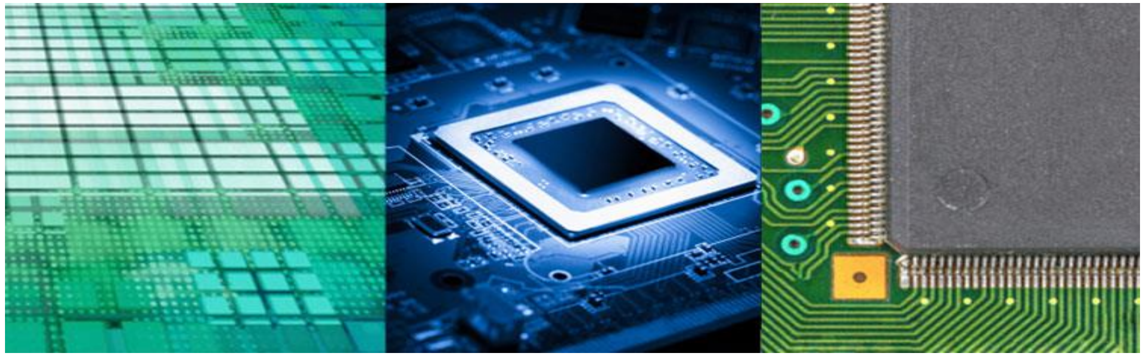
Agenda

Multi-target Production Code Generation

Hardware targets

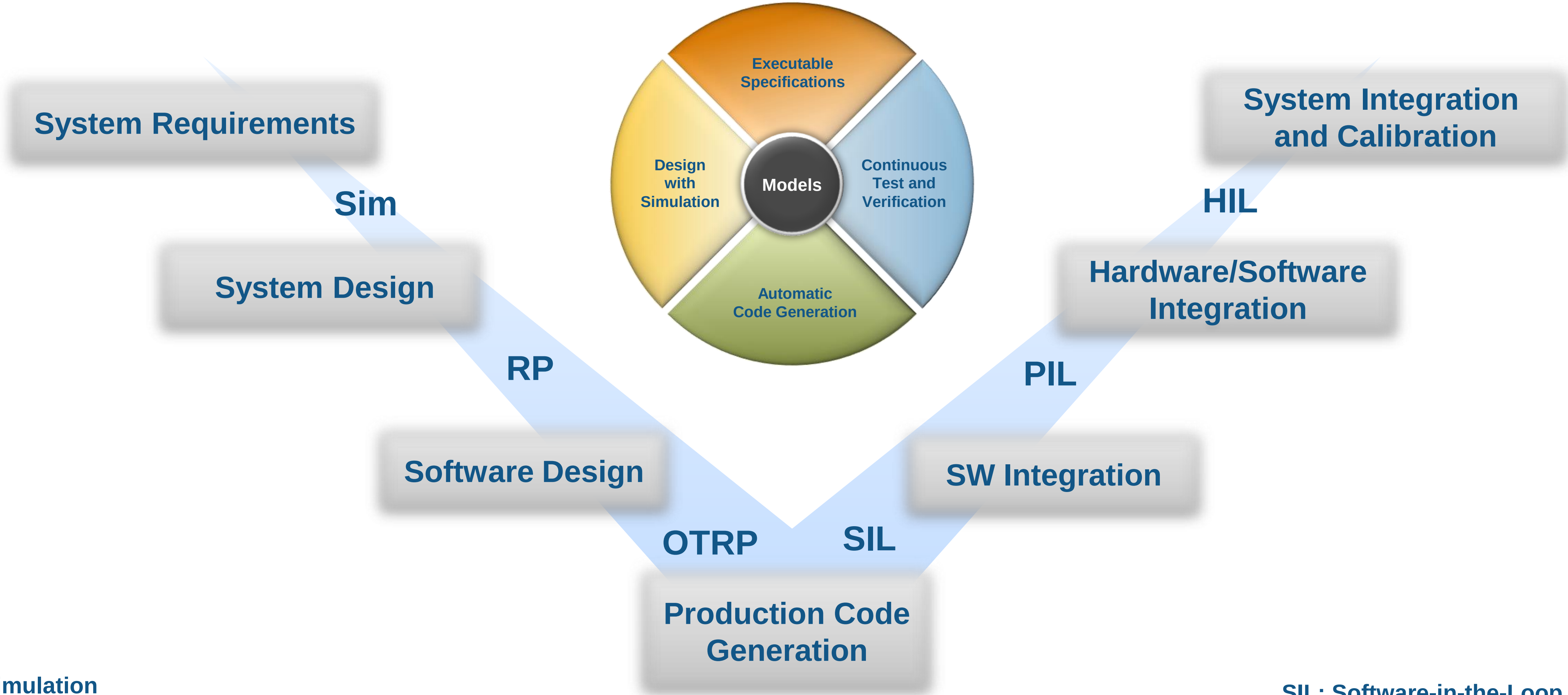
Continuous Verification and Validation

Summary



Embedded System Development Process

with Model-Based Design



System Requirements

Sim

System Design

RP

Software Design

OTRP

Production Code Generation

SIL

SW Integration

PIL

Hardware/Software Integration

HIL

System Integration and Calibration

Executable Specifications

Design with Simulation

Models

Continuous Test and Verification

Automatic Code Generation

Sim: Simulation

RP: Rapid Prototyping

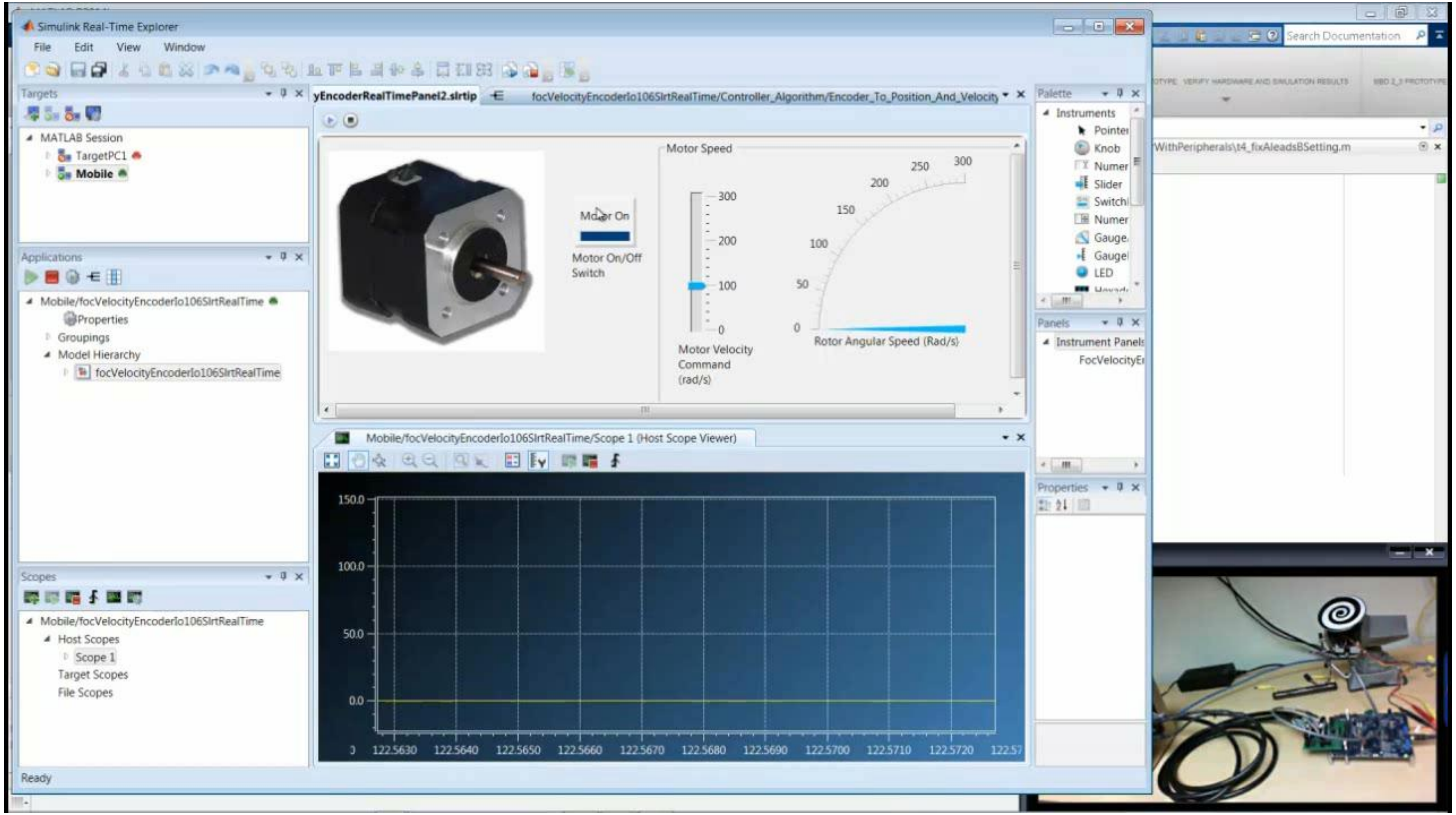
OTRP: On-Target Rapid Prototyping

SIL: Software-in-the-Loop Testing

PIL: Processor-in-the-Loop Testing

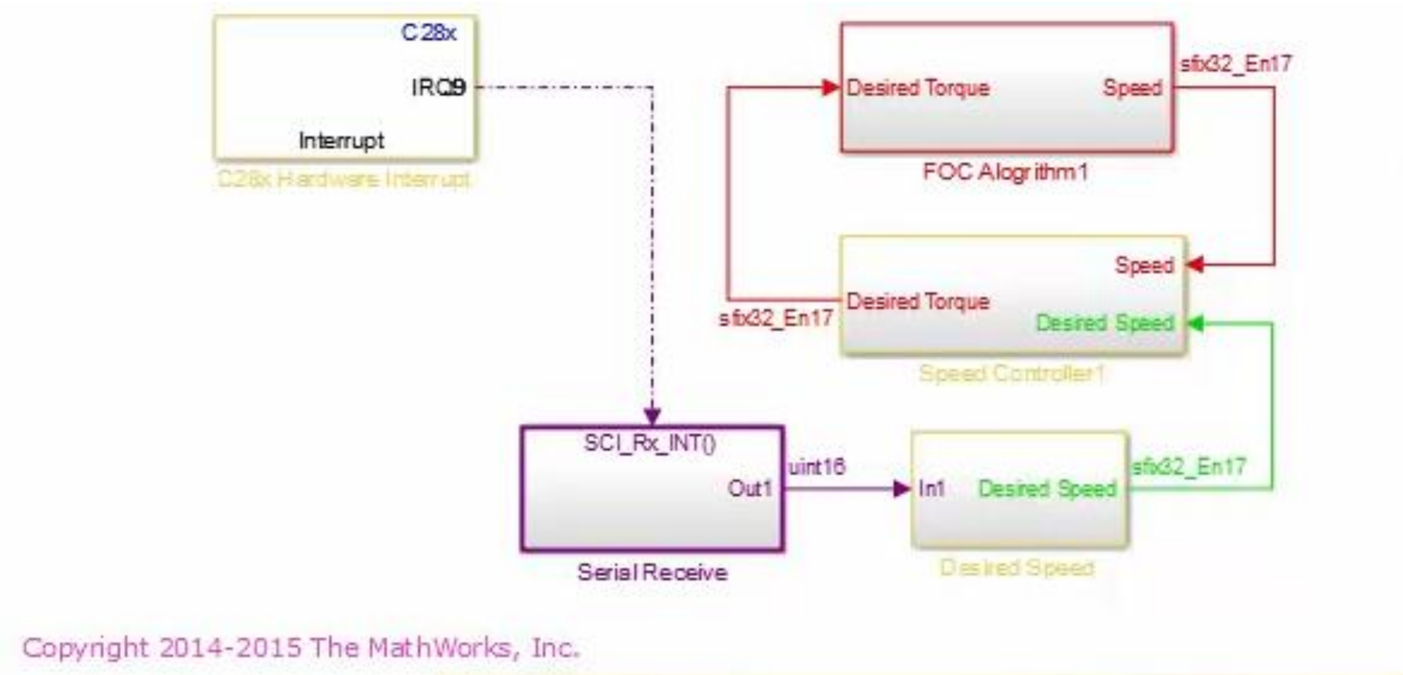
HIL: Hardware-in-the-Loop Testing

Rapid Prototyping

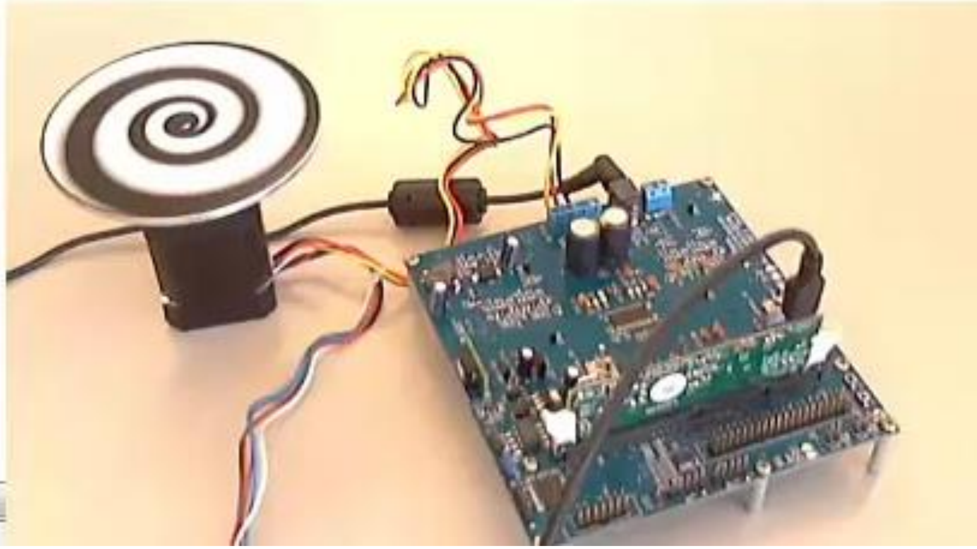


On-Target Prototyping

- Does algorithm perform well on actual device with true latencies?

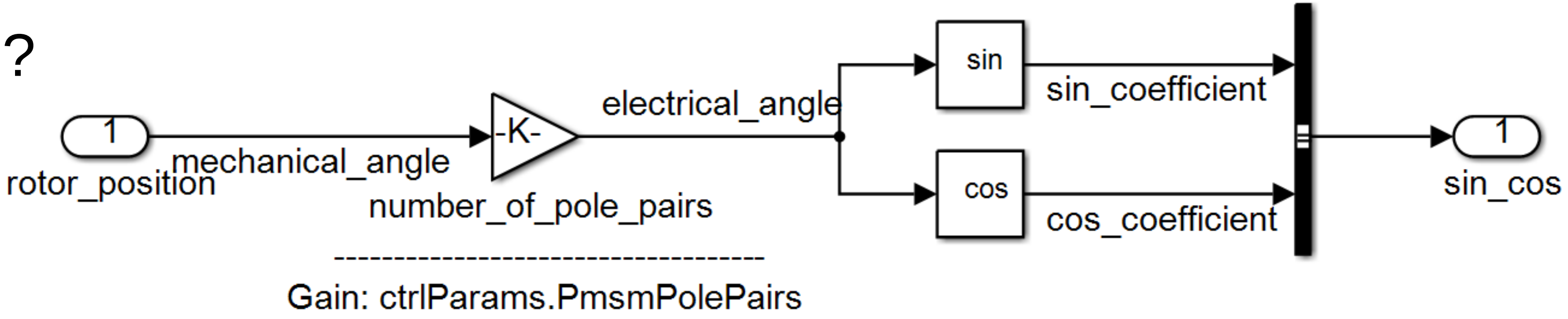


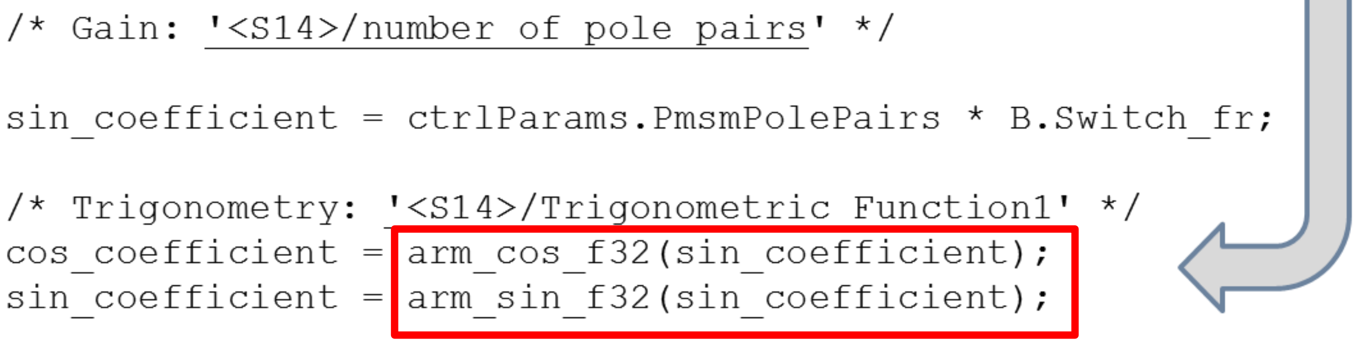
Copyright 2014-2015 The MathWorks, Inc.



Production Code Generation

- Is the code optimized?



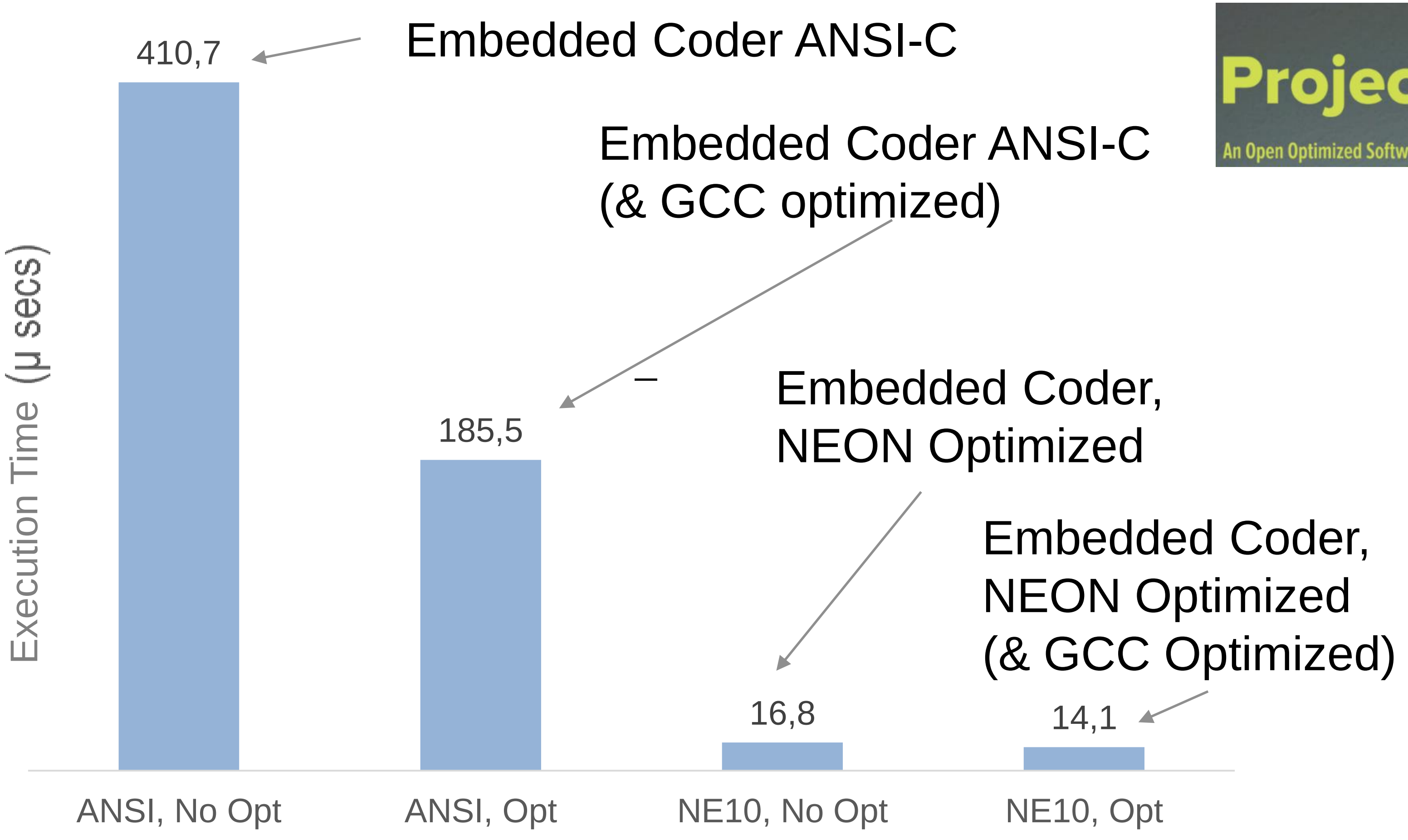


```
/* Gain: '<S14>/number of pole pairs' */

sin_coefficient = ctrlParams.PmsmPolePairs * B.Switch_fr;

/* Trigonometry: '<S14>/Trigonometric Function1' */
cos_coefficient = arm_cos_f32(sin_coefficient);
sin_coefficient = arm_sin_f32(sin_coefficient);
```

Embedded Coder®

- Code Replacement Tables
- Use of e.g. CMSIS library for code optimization for ARM

Results for ARM Cortex-A



Embedded Coder ANSI-C

Embedded Coder ANSI-C
(& GCC optimized)

Embedded Coder,
NEON Optimized

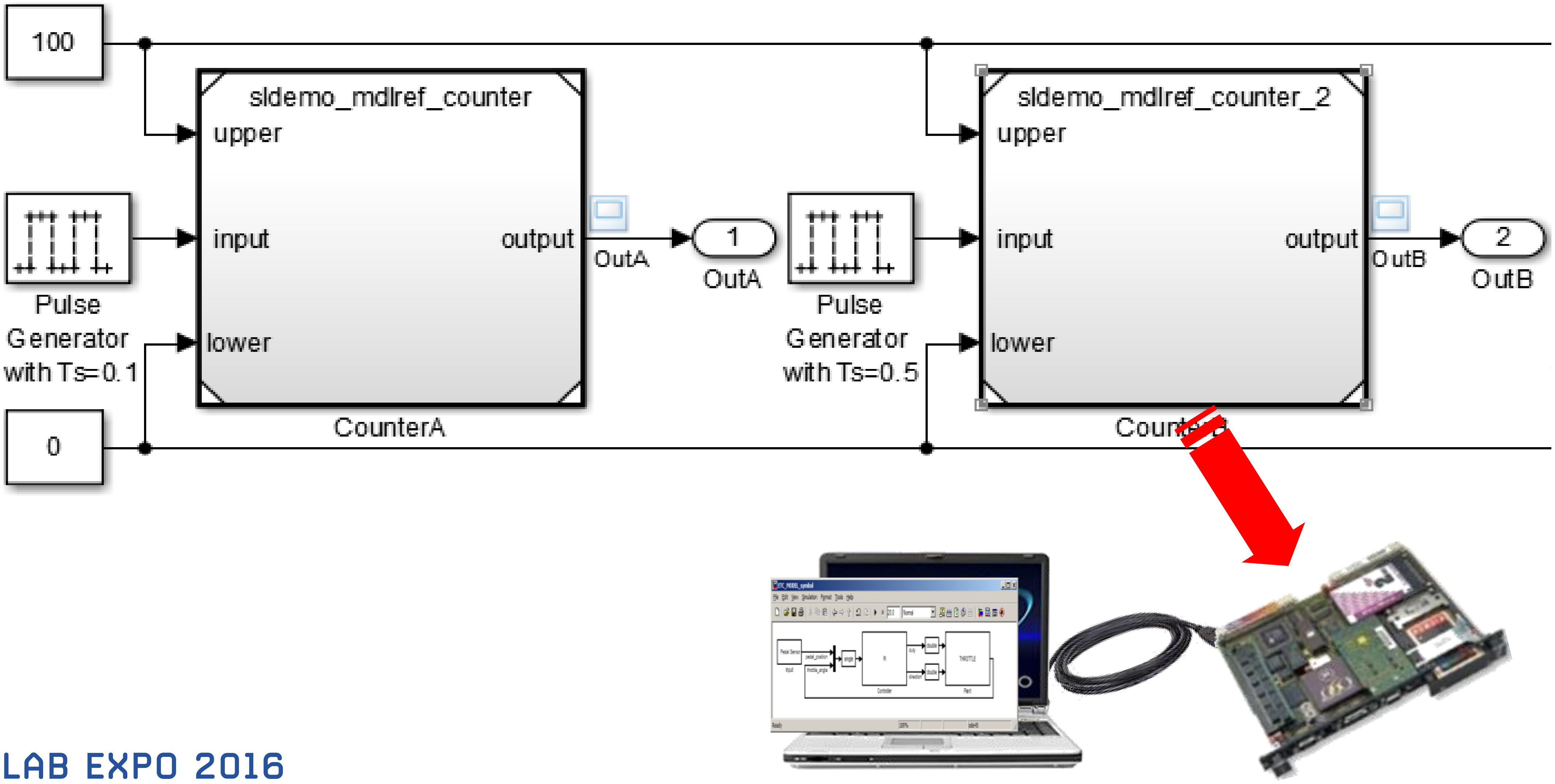
Embedded Coder,
NEON Optimized
(& GCC Optimized)

Project Ne10

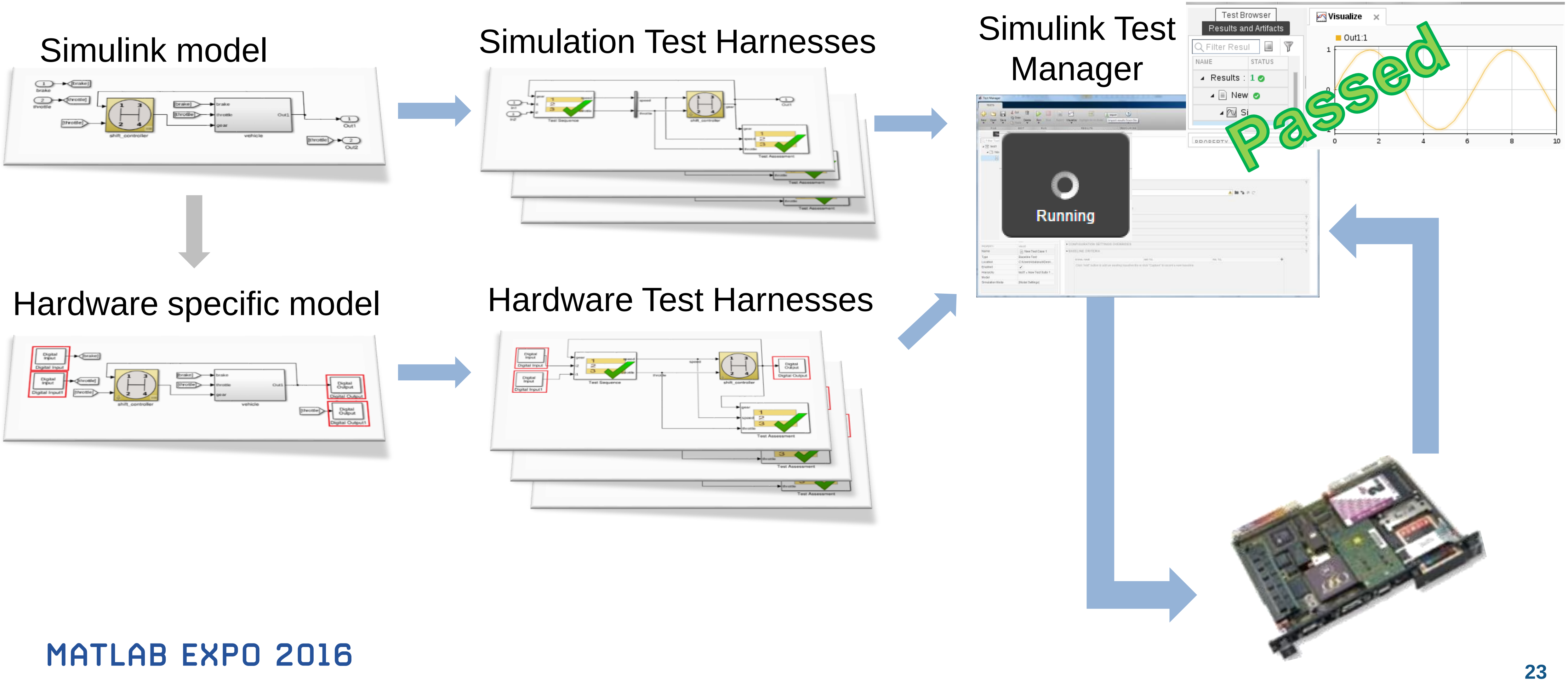
An Open Optimized Software Library Project for the ARM Architecture

Run Format: [ANSI or Ne10], [gcc no opt or gcc -02], ARM 1Ghz Cortex A8

Processor-in-the-loop



Target requirement-based testing



Simulink model

Simulation Test Harnesses

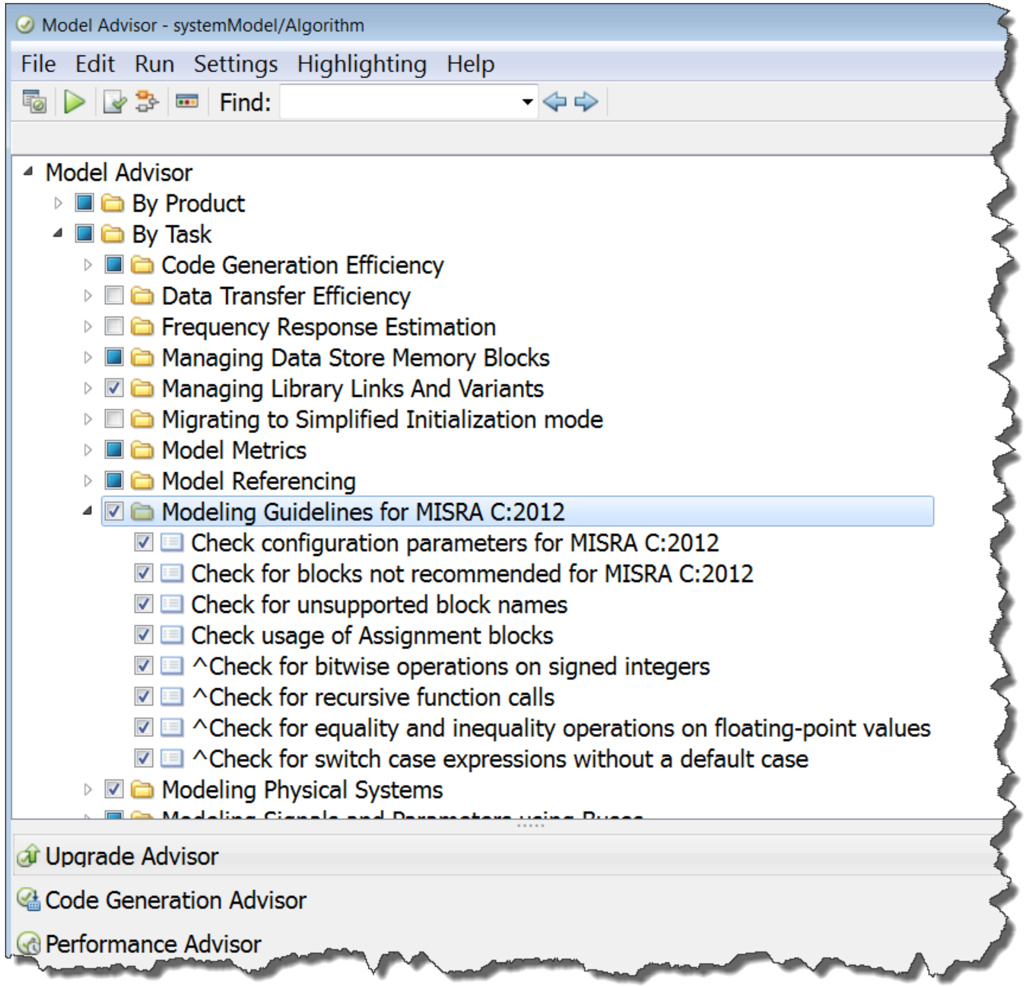
Simulink Test Manager

Hardware specific model

Hardware Test Harnesses

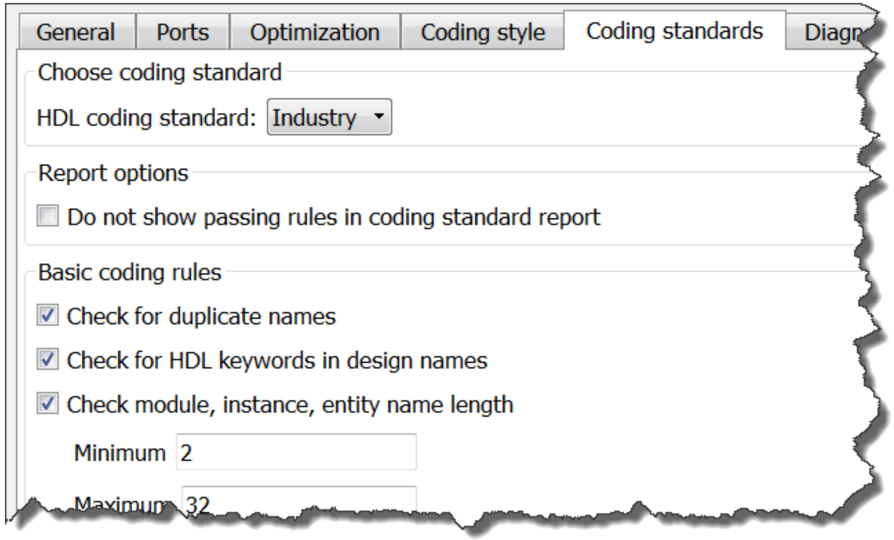
Coding Standards

MISRA-C


Model Advisor - systemModel/Algorithm
File Edit Run Settings Highlighting Help
Find:
Upgrade Advisor
Code Generation Advisor
Performance Advisor

- Model Advisor
  - ■ By Product
  - ■ By Task
    - ■ Code Generation Efficiency
    - □ Data Transfer Efficiency
    - □ Frequency Response Estimation
    - ■ Managing Data Store Memory Blocks
    - ☑ Managing Library Links And Variants
    - □ Migrating to Simplified Initialization mode
    - ■ Model Metrics
    - ■ Model Referencing
    - ☑ Modeling Guidelines for MISRA C:2012
      - ☑ Check configuration parameters for MISRA C:2012
      - ☑ Check for blocks not recommended for MISRA C:2012
      - ☑ Check for unsupported block names
      - ☑ Check usage of Assignment blocks
      - ☑ ^Check for bitwise operations on signed integers
      - ☑ ^Check for recursive function calls
      - ☑ ^Check for equality and inequality operations on floating-point values
      - ☑ ^Check for switch case expressions without a default case
    - ☑ Modeling Physical Systems
    - ■ Modeling Signals and Parameters using Buses

STARC HDL



Choose coding standard

HDL coding standard: Industry

Report options

□ Do not show passing rules in coding standard report

Basic coding rules

☑ Check for duplicate names

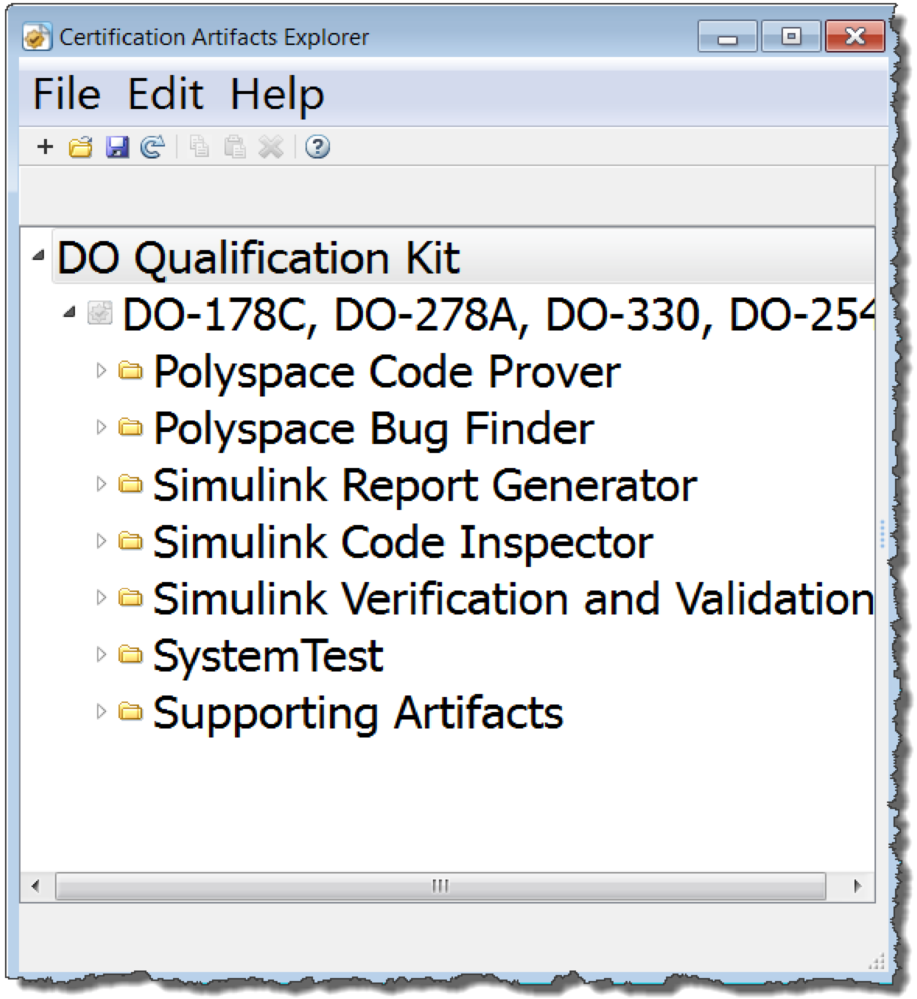
☑ Check for HDL keywords in design names

☑ Check module, instance, entity name length

Minimum 2

Maximum 32

DO Qualification Kit



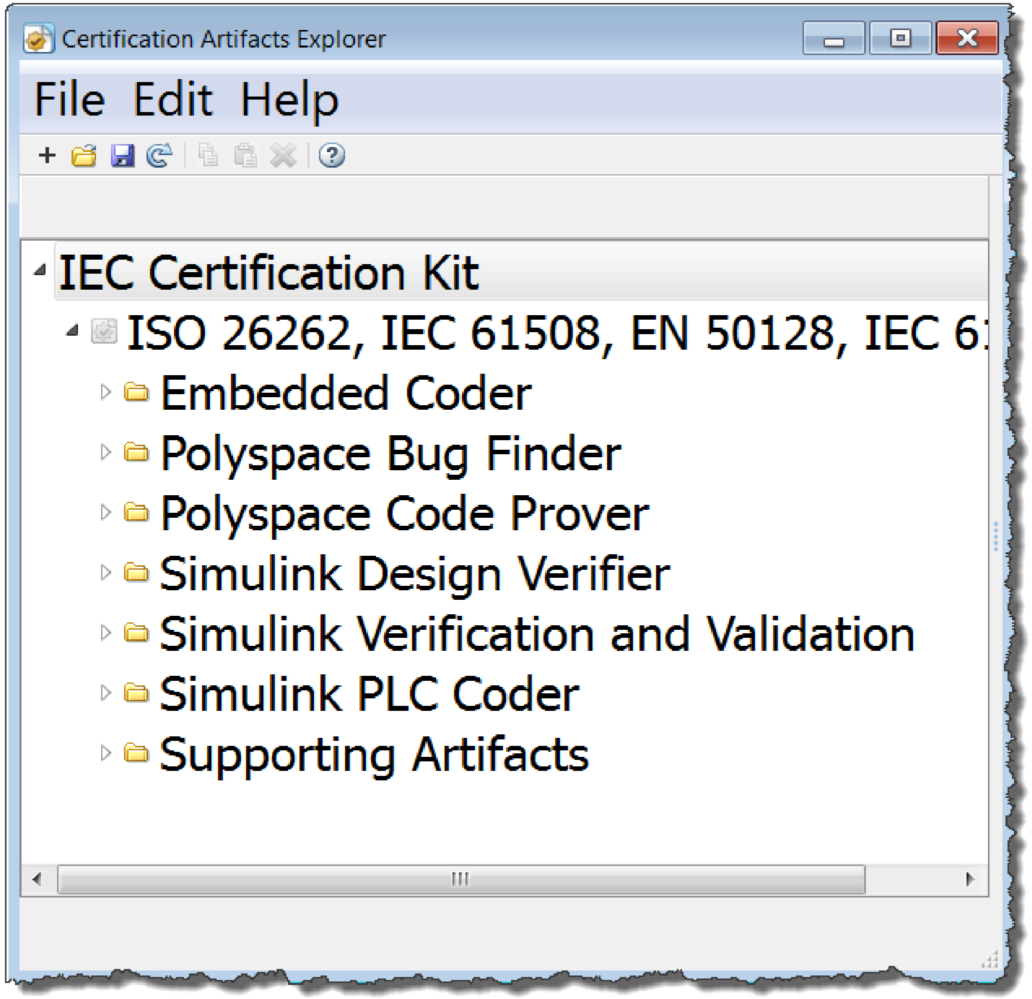
Certification Artifacts Explorer

File Edit Help

- DO Qualification Kit
  - DO-178C, DO-278A, DO-330, DO-254
    - Polyspace Code Prover
    - Polyspace Bug Finder
    - Simulink Report Generator
    - Simulink Code Inspector
    - Simulink Verification and Validation
    - SystemTest
    - Supporting Artifacts

```
>>qualkitdo
```

IEC/ISO Certification Kit



Certification Artifacts Explorer

File Edit Help

- IEC Certification Kit
  - ISO 26262, IEC 61508, EN 50128, IEC 6
    - Embedded Coder
    - Polyspace Bug Finder
    - Polyspace Code Prover
    - Simulink Design Verifier
    - Simulink Verification and Validation
    - Simulink PLC Coder
    - Supporting Artifacts

```
>>certkitiec
```

Model-Based Design – Certification Examples

DO-178 (Level A)

Honeywell Aerospace USA
Flight Control Systems

ISO 26262

TRW Germany
Electronic parking brake control system

ARP4754 & DO-178

Airbus Helicopters
Certified flight software

IEC 62304

Weinmann Medical Germany
Transport ventilator

IEC 61508

Alstom Grid UK
HDVC Power Systems

EN 50128

Alstom France
Train Control Systems

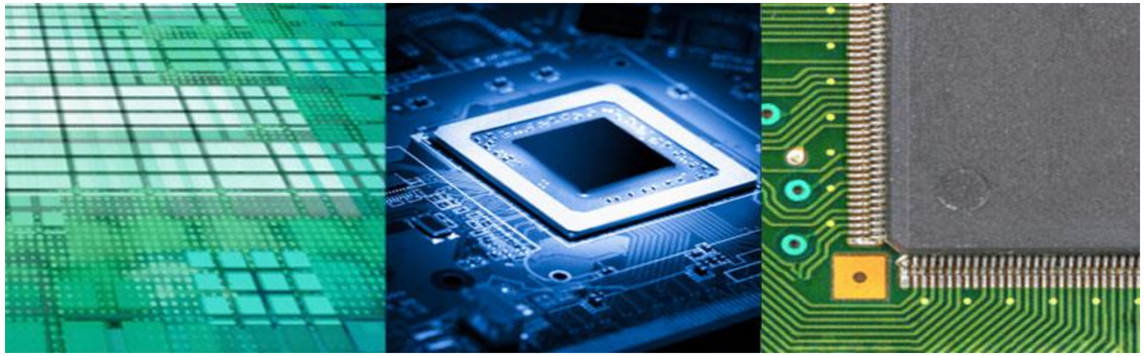
Agenda

Multi-target Production Code Generation

Hardware targets

Continuous Verification and Validation

Summary



Key Take-Aways

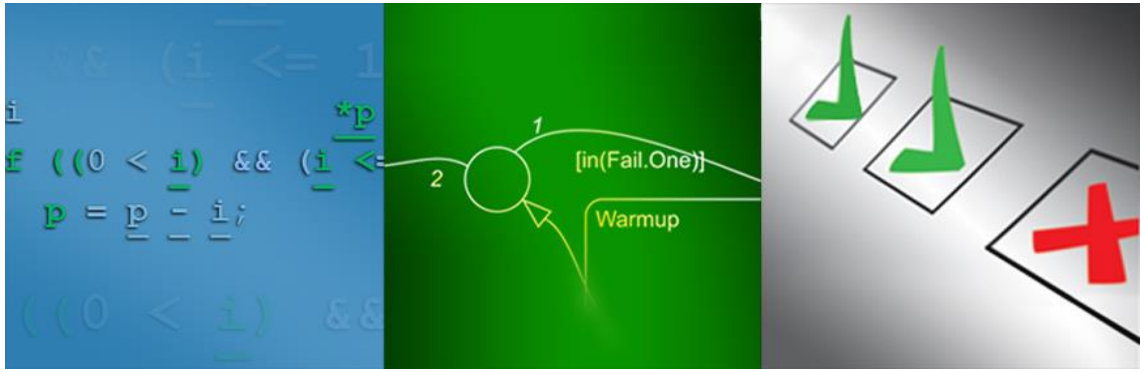
Code generation is expanding rapidly

- C
- C++

- VHDL
- Verilog

- Structured Text

Code generation offers many benefits



Hardware resources need optimization