

MATLAB EXPO

Certification of a Flight Control System Implemented on an SoC

Bill Potter



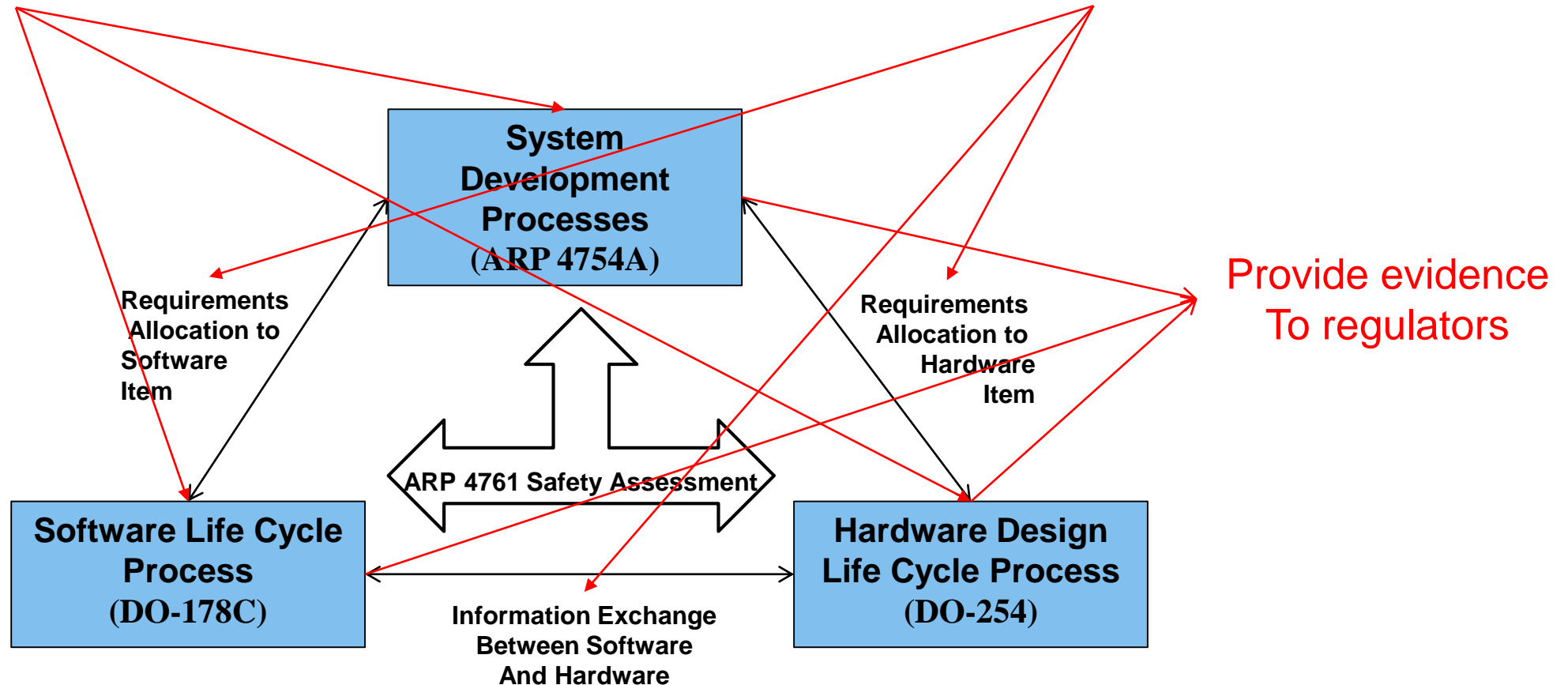
Key Takeaways

1. Model-Based Design increases productivity for development of certified systems
2. A single development environment for System on a Chip
3. Qualified tools that span ARP 4754A, DO-178C and DO-254

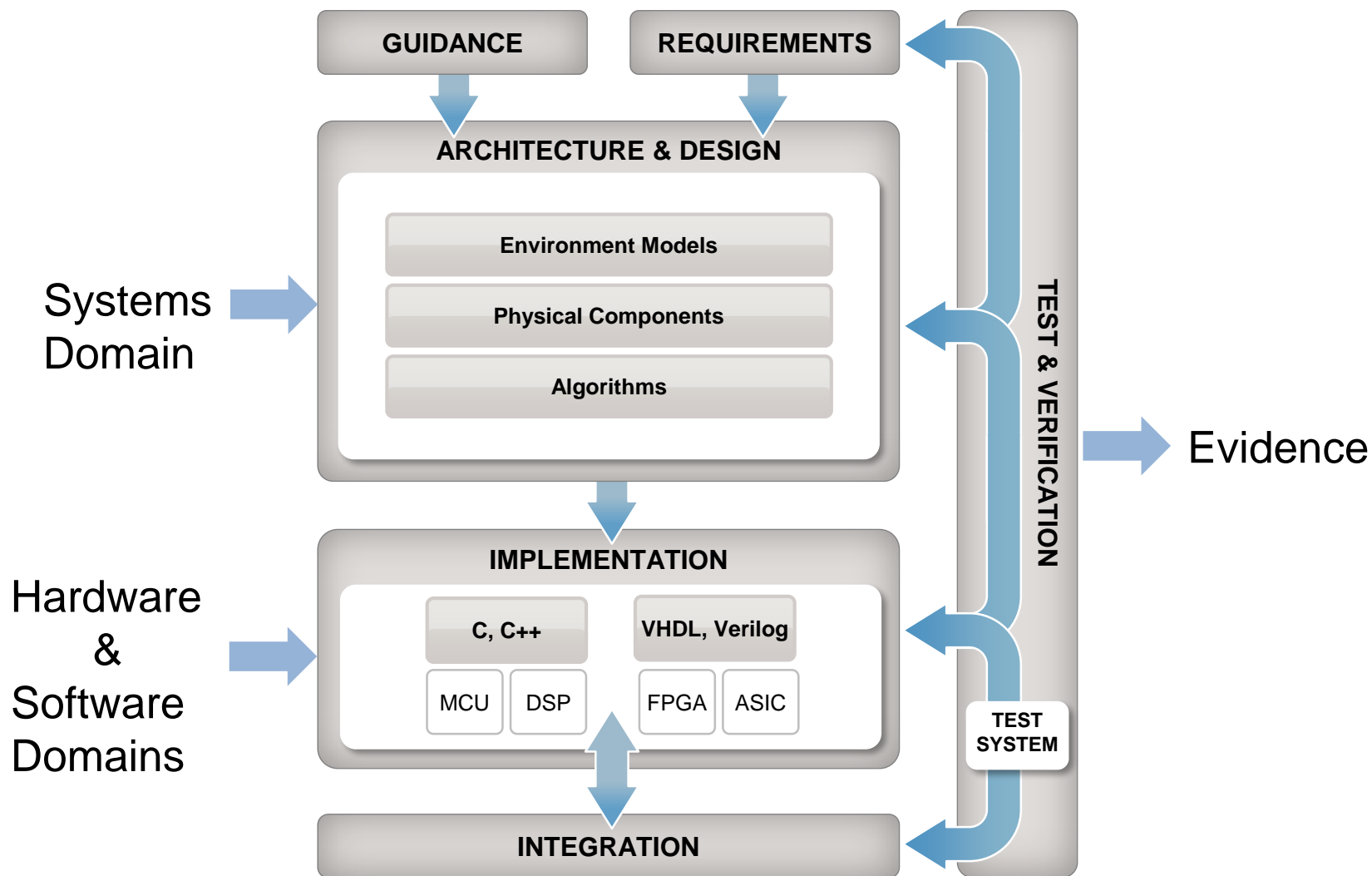
Certification is Difficult

Comply with regulatory guidance

Coordinate between engineering domains



Solution - Model-Based Design

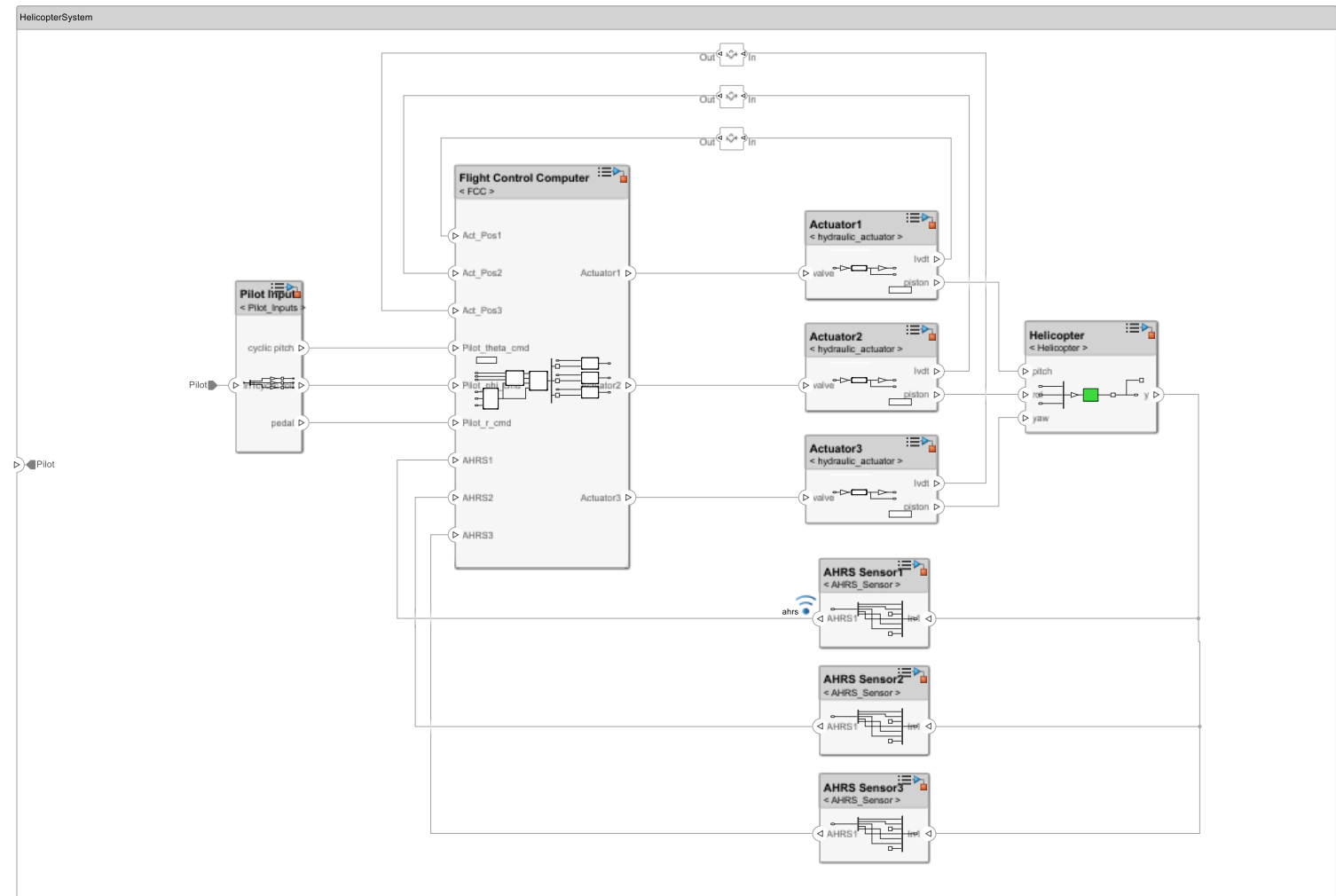


System Model Development

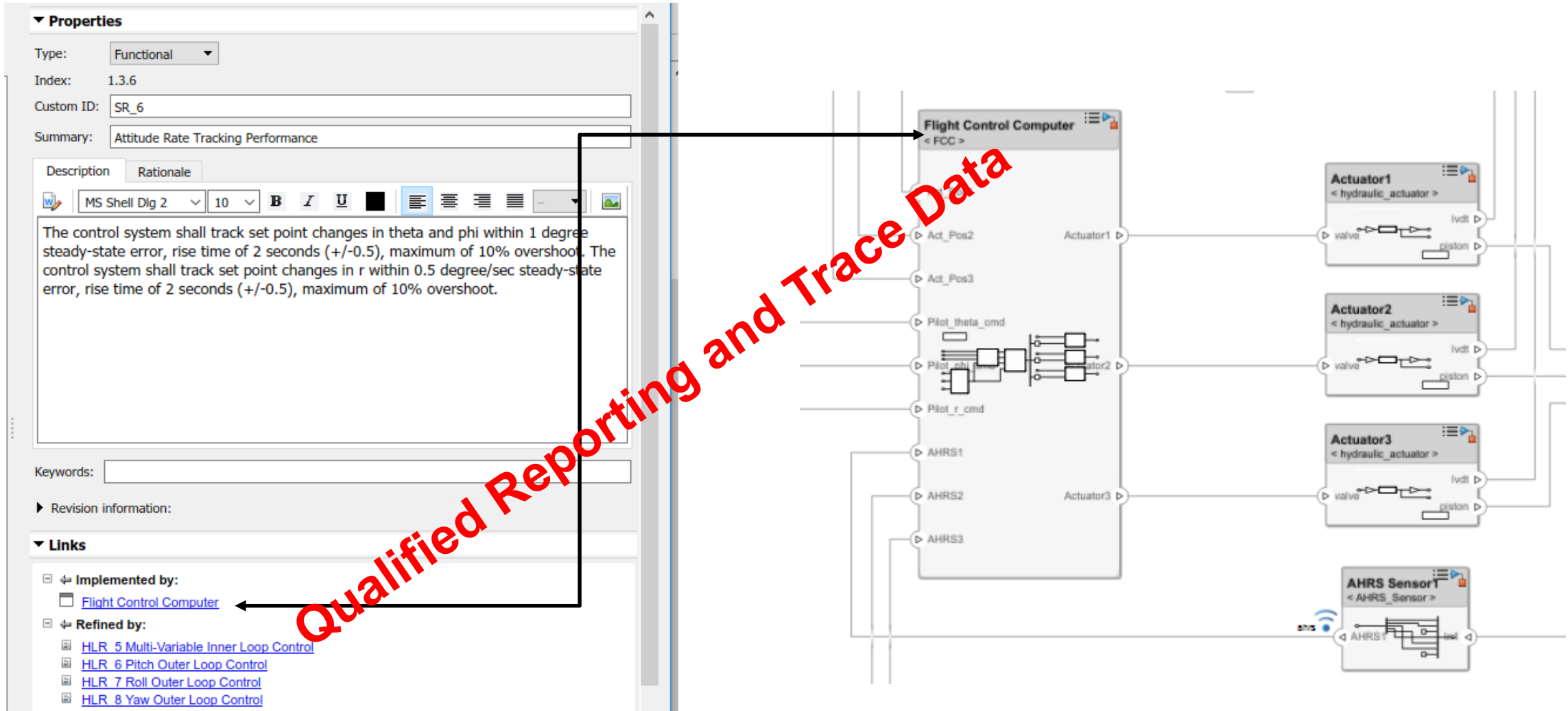
System Level Architecture Model of Helicopter Flight Controls

System Architecture Components

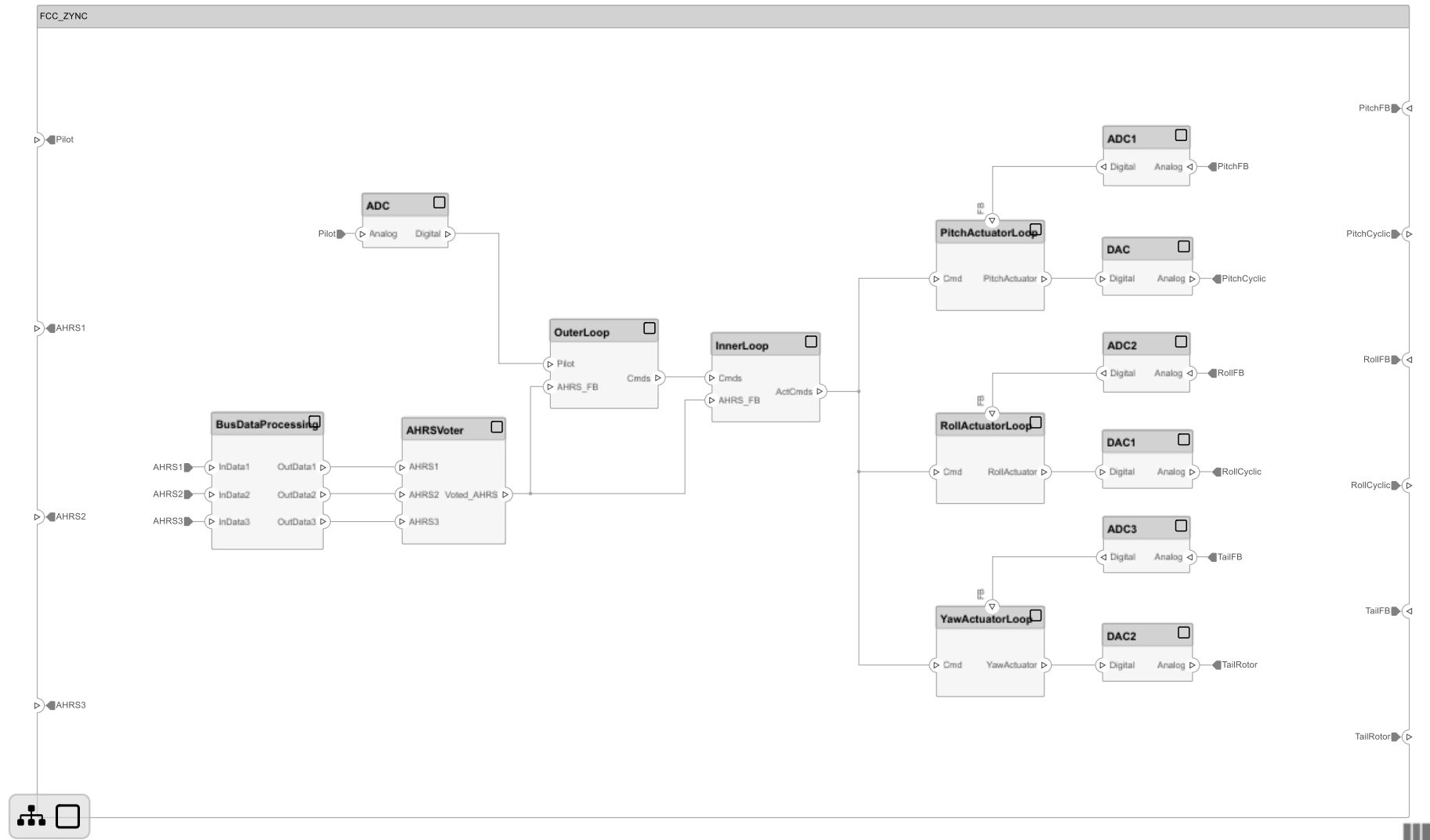
- Plant Models
- Physical Items
- Digital Hardware Items
- Software Items
- Redundancy



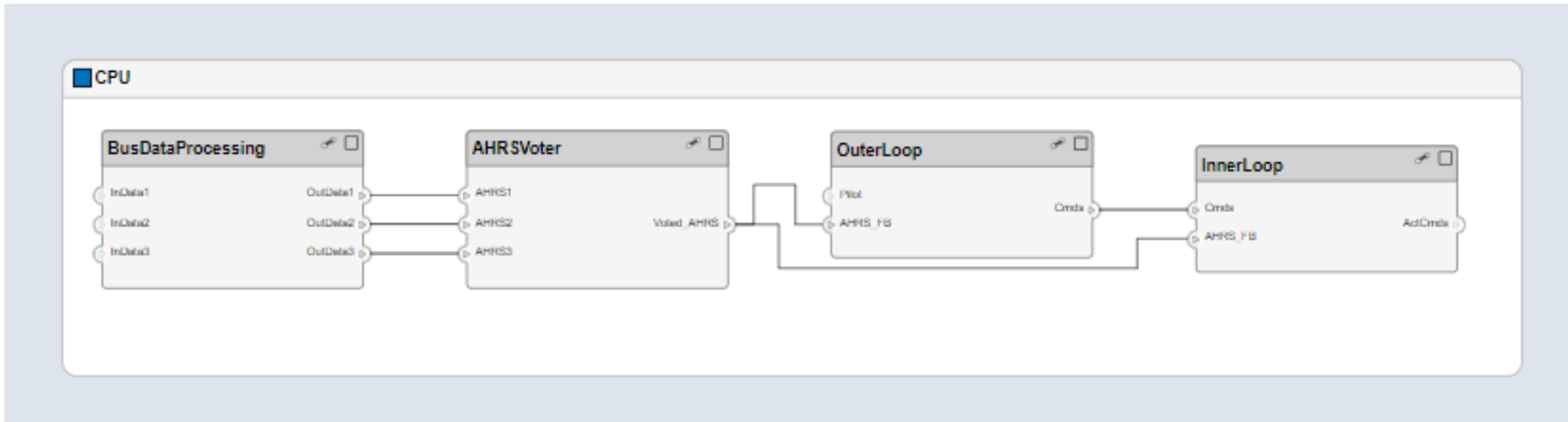
System Requirements to Architecture Traceability



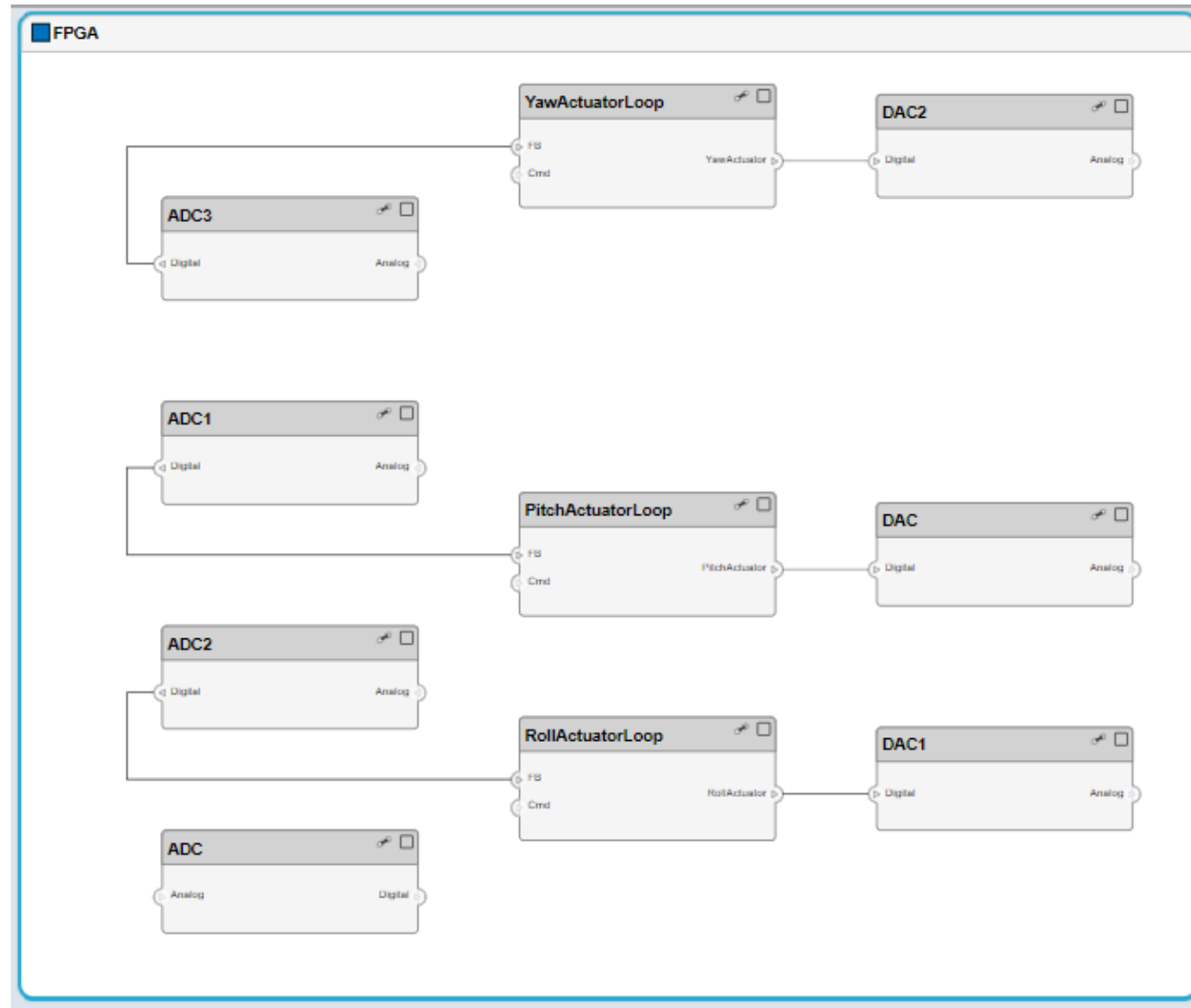
Flight Control Computer Architecture for Xilinx Zynq SoC



Allocation of System Requirements to Software – CPU View



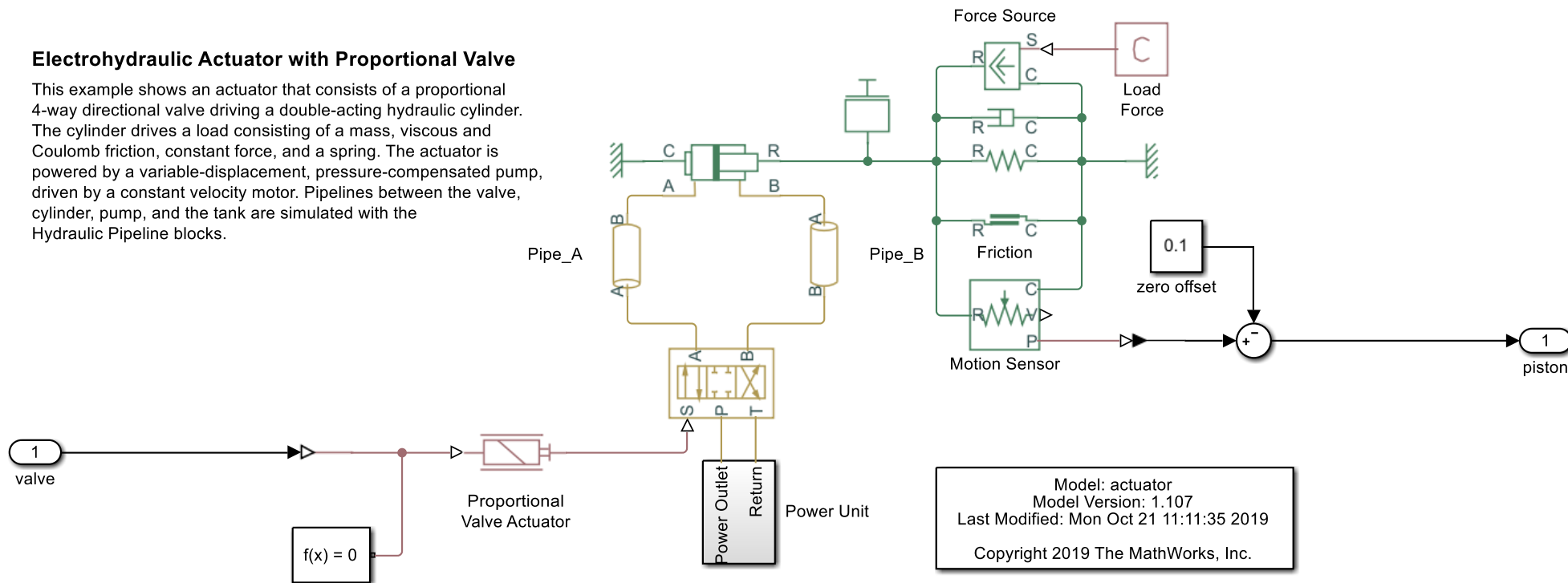
Allocation of System Requirements to Hardware – FPGA View



Plant Models can be used with System Composer Components

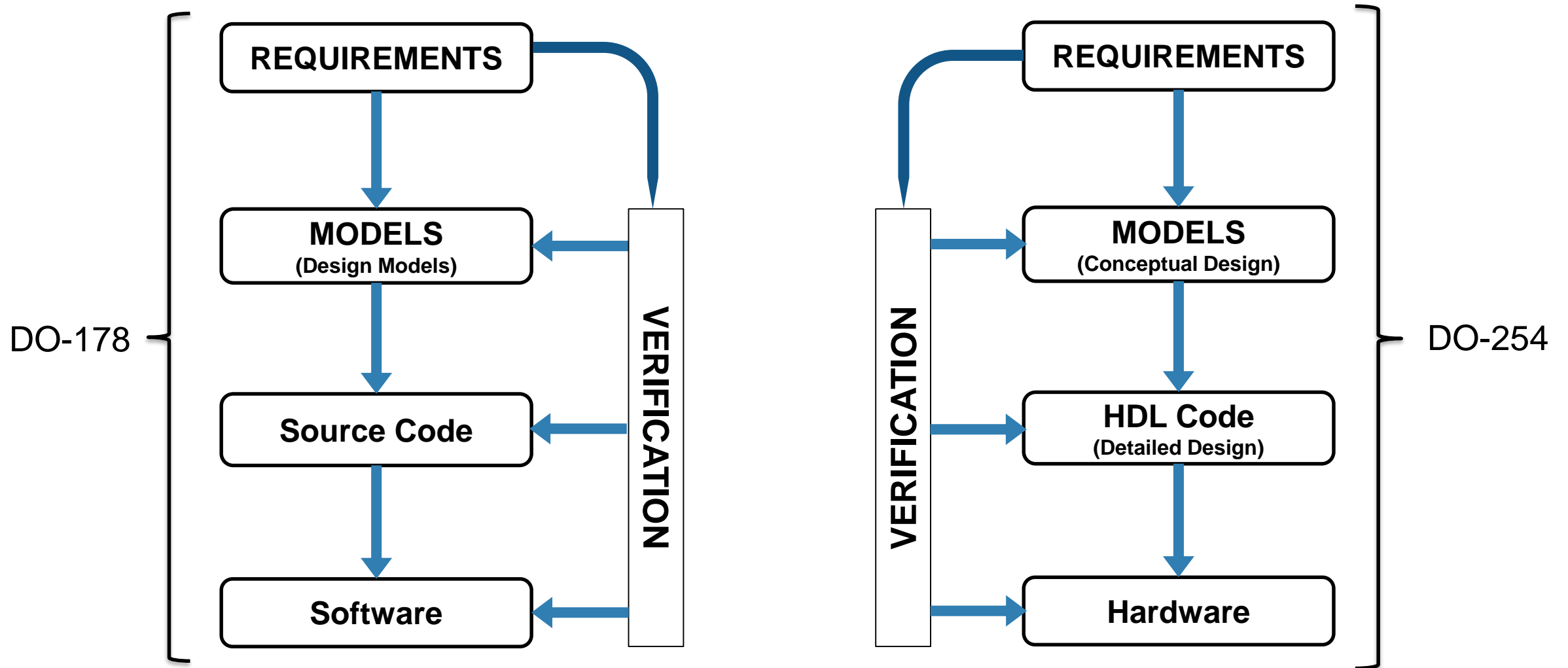
Electrohydraulic Actuator with Proportional Valve

This example shows an actuator that consists of a proportional 4-way directional valve driving a double-acting hydraulic cylinder. The cylinder drives a load consisting of a mass, viscous and Coulomb friction, constant force, and a spring. The actuator is powered by a variable-displacement, pressure-compensated pump, driven by a constant velocity motor. Pipelines between the valve, cylinder, pump, and the tank are simulated with the Hydraulic Pipeline blocks.



Software and Hardware Model Development

DO-178 and DO-254 Processes Using Models



Trace Models to Requirements

ActuatorLoop - Simulink

Model Name: ActuatorLoop
Model Version: 1.34
Last Modified: Sat Jan 25 12:28:13 2020
Copyright 2019 The MathWorks, Inc.

Sub3 & Sub4 implement anti-windup of integrator

Property Inspector

Requirement: HR_3

Details

▼ Properties

Type: Functional

Index: 1.3.3

Custom ID: HR_3

Summary: Hydraulic Actuator Loop Control

Description Rationale

Each hydraulic actuator loop shall be implemented as a proportional/integral/derivative (PID) control loop operating at 1 millisecond frame rate.

The proportional gain shall be 0.135.
The integral gain shall be 0.00122.
The derivative gain shall be -0.00134.
The derivative filter coefficient shall be 0.0157

Keywords:

► Revision information:

▼ Links

Derived from:
[SR_4 Hydraulic Actuator Control Loop](#)

Implemented by:
[Gain](#)

Requirements - ActuatorLoop

View: Requirements

Index	ID	Summary
1.3	High-Level Hardware Requirements	High-Level Hardware Requirements
1.3.1	HR_1	Hydraulic Actuator Feedback
1.3.2	HR_2	Hydraulic Actuator Drive
1.3.3	HR_3	Hydraulic Actuator Loop Control
1.3.4	HR_4	Hydraulic Actuator Command

Ready 79% FixedStepDiscrete

Verify Conformance to Model Standards

The screenshot displays the MATLAB Simulink environment with the Model Advisor tool open. The Model Advisor window shows a tree view of checks for DO-254 standards, including Display model version information, High-Integrity Systems, Library Links, Model Referencing, Requirements Consistency, HDL Code, Checks for blocks and block settings, and Industry standard checks. The Industry standard checks are expanded, showing a list of checks with green checkmarks indicating they are passed. A red arrow points from the Model Advisor icon in the top toolbar to the Model Advisor window. A large red diagonal watermark text "Qualified Conformance to Standards" is overlaid across the center of the image.

Model Advisor - ActuatorLoop C:\Users\bpotter\MATLAB\Projects\HDL_Example_Project\tools\checks\DO254_Checks.mat

File Edit Run Settings Highlighting Help

Find: [] [] []

Model Advisor

- By Task
 - Modeling Standards for DO-254
 - ^Display model version information
 - High-Integrity Systems
 - Library Links
 - Model Referencing
 - Requirements Consistency
 - HDL Code
 - Checks for blocks and block settings
 - Industry standard checks
 - Check VHDL file extension
 - Check naming conventions
 - Check top-level subsystem port names
 - Check module/entity names
 - Check signal and port names
 - Check package file names
 - Check generics
 - Check clock, reset, and enable signals
 - Check architecture name
 - Check entity and architecture
 - Check clock settings
 - Model configuration checks
 - Native Floating Point checks
 - Checks for ports and subsystems

Modeling Standards for DO-254

Model Advisor Analysis

contains checks for safety critical or mission critical code generation

Run Selected Checks

☐ Show report after run

Report

Report: Generate Report... \report_2.html

Date/Time: 04-Feb-2020 06:54:44

Summary: ✔ Pass: 89 ✘ Fail: 0 ⚠ Warning: 1 📄 Not Run: 0

Tips

To process all enabled items in this folder and generate a new report, click "Run Selected Checks".

Right-click to select or deselect all items in this folder.

To automatically display the report after processing, select "Show report after run".

To display the last report generated, click the "Report" path link.

For a list of all possible actions, right-click an item in the Task Hierarchy.

To show or hide By Product folder, select or clear "Show By Product Folder" in the Settings > Preferences dialog box.

Help

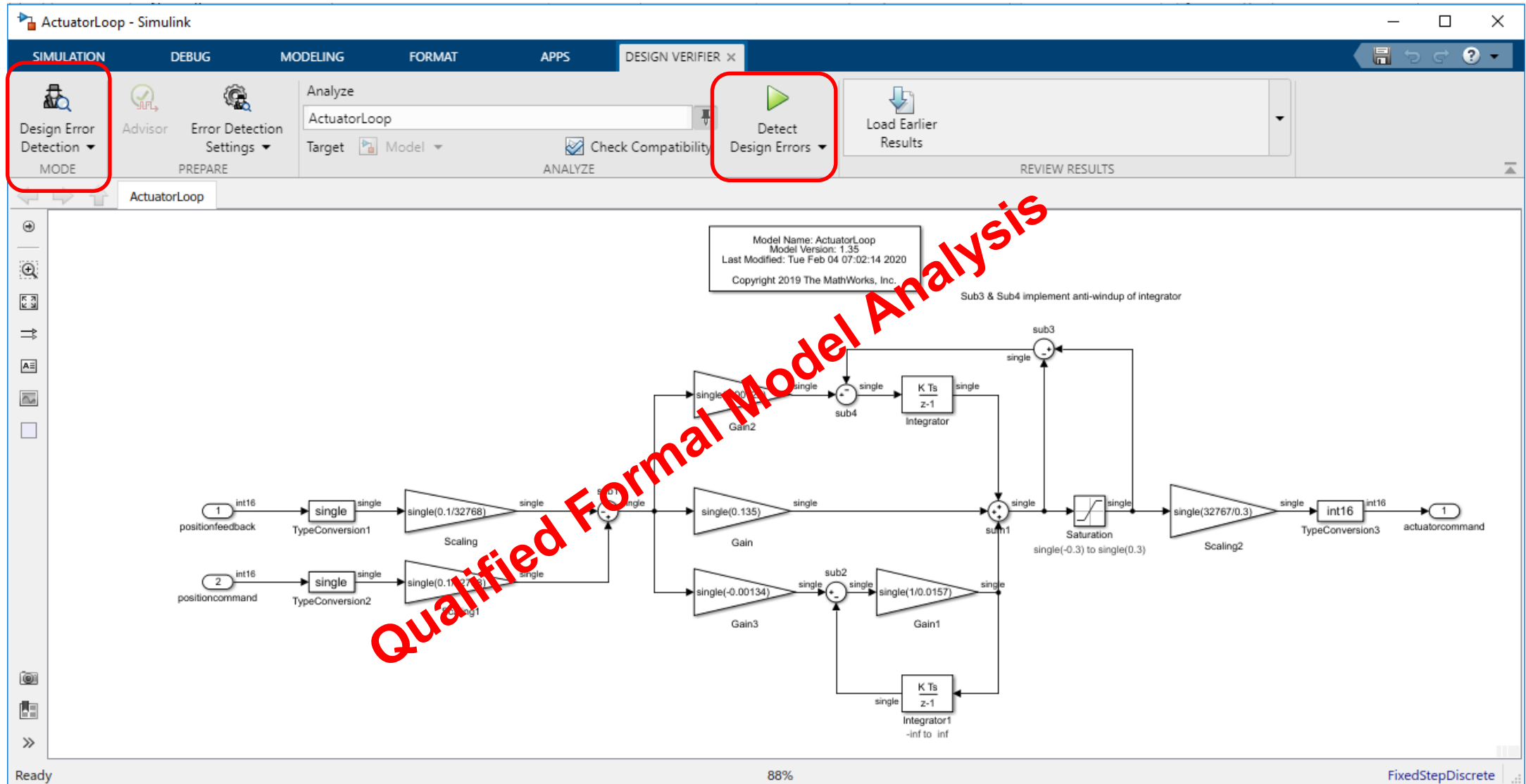
1 int16 positionfeedback → single TypeConversion1 → single

2 int16 positioncommand → single TypeConversion2 → single

1 actuatorcommand

Ready 109% FixedStepDiscrete

Perform Formal Analysis on Models



Integrated Requirements, Simulation & Model Coverage Analysis

The image illustrates the integrated workflow for requirements, simulation, and model coverage analysis in MATLAB/Simulink. A large red diagonal watermark reads "Qualified Pass/Fail Checking and Coverage".

Top Left: Simulink Main Window

- Menu: SIMULATION, DEBUG, MODELING, FORMAT, APPS
- Buttons: Get Add-Ons, Requirements Manager, Simulink Test, Coverage Analyzer, Model Advisor
- Environment: ActuatorLoop

Top Right: Coverage Analyzer Window

- Menu: SIMULATION, DEBUG, MODELING, FORMAT, APPS, COVERAGE, DESIGN VERIFIER
- Buttons: Coverage ON, Cumulative Collection, Settings, Step Time (10.0), Normal, Step Back, Analyze Coverage, Step Forward, Stop, Results Explorer, Load Earlier Results

Bottom Left: Requirements Editor Window

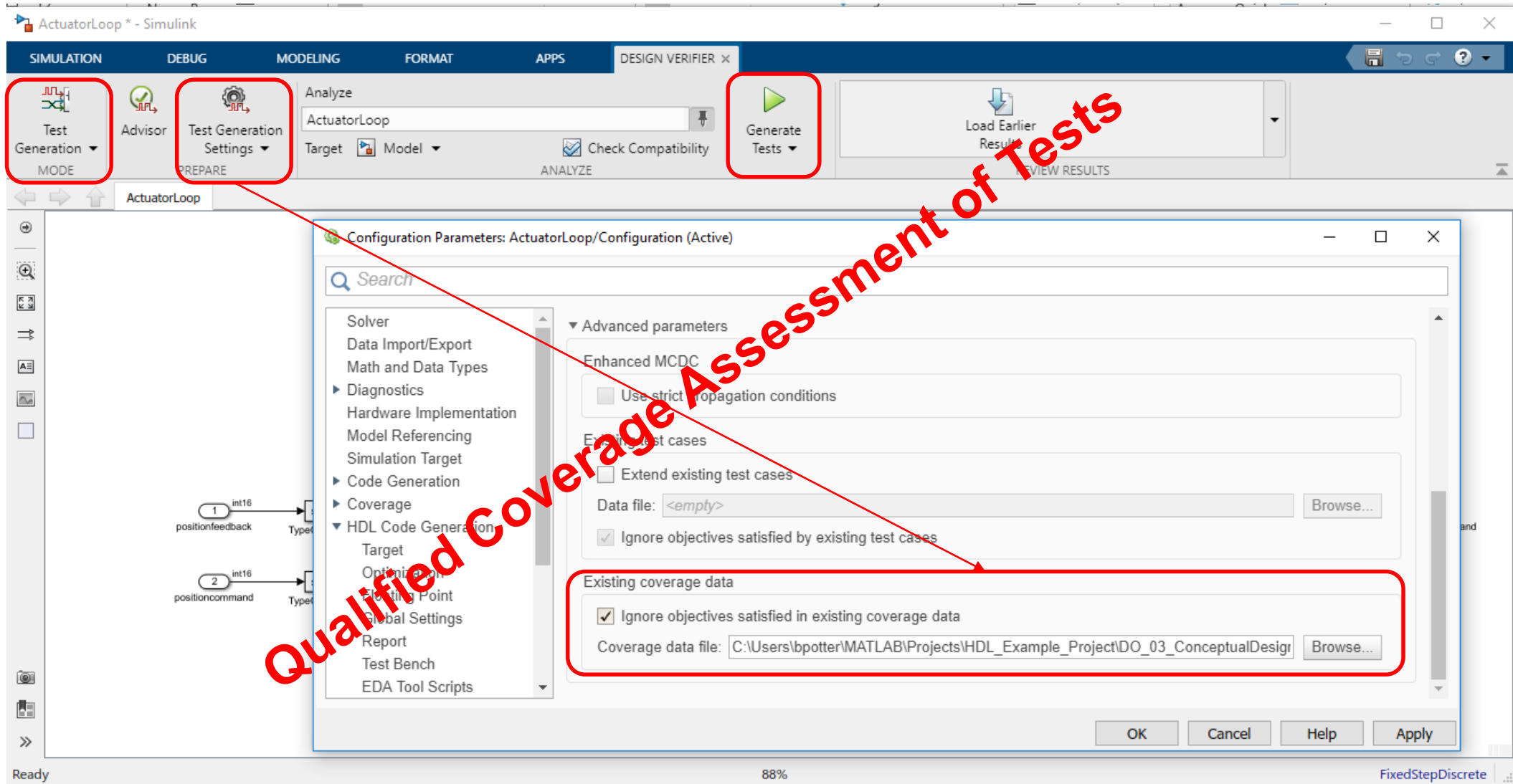
- Menu: File, Edit, Display, Analysis, Report, Help
- View: Requirements
- Table:

Index	ID	Summary	Implemented	Passed
HelicopterHardwareRequirements				
1	HelicopterHardwareRequirements	Requirements for an FPGA actuator lo...		
1.1	Introduction	Introduction		
1.2	System Description	System Description		
1.3	High-Level Hardware Requirements	High-Level Hardware Requirements		
1.3.1	HR_1	Hydraulic Actuator Feedfor...		
1.3.2	HR_2	Hydraulic Actuator Feedfor...		
1.3.3	HR_3	Hydraulic Actuator Loop Control		
1.3.4	HR_4	Hydraulic Actuator Command		
HelicopterSystemRequirements				

Bottom Right: Test Manager Window

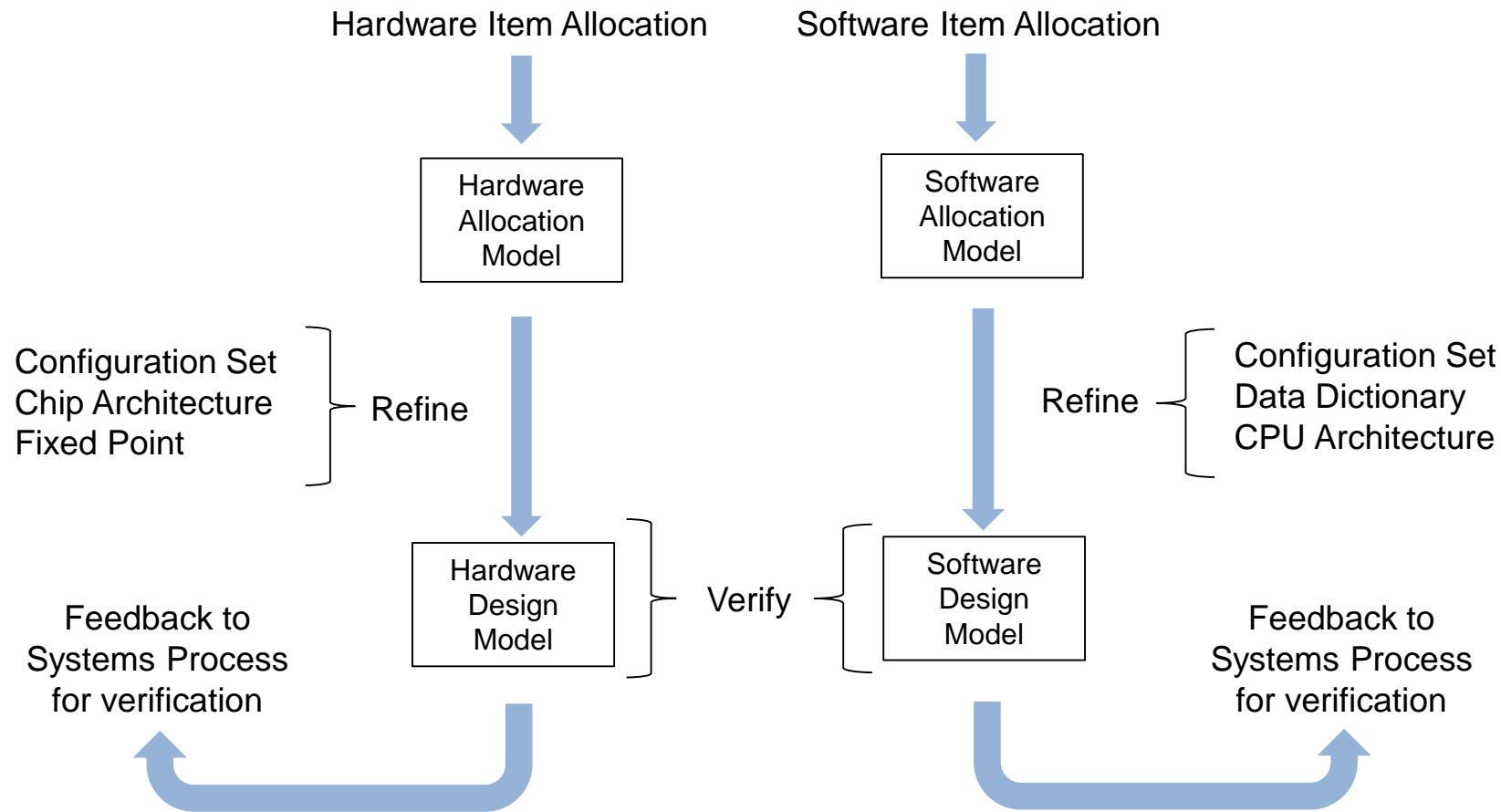
- Menu: FILE, EDIT, RUN, RESULTS, ENVIRONMENT, RESOURCES
- Test Browser: Filter tests by name or tags, e.g. tags: test
- Test: ChirpFrequencyResponse
- Test Properties: Name, Type, Model, Simulation Mode, Location, Enabled, Hierarchy
- Test Results: ChirpFrequencyResponse, Simulation Test, Select releases for simulation: Current, Create Test Case from External File, TAGS, DESCRIPTION*, REQUIREMENTS*, SYSTEM UNDER TEST*, Model: ActuatorControlLoop, TEST HARNESS, SIMULATION SETTINGS OVERRIDES*, PARAMETER OVERRIDES, CALLBACKS*, INPUTS

Test Generation for Missing Coverage



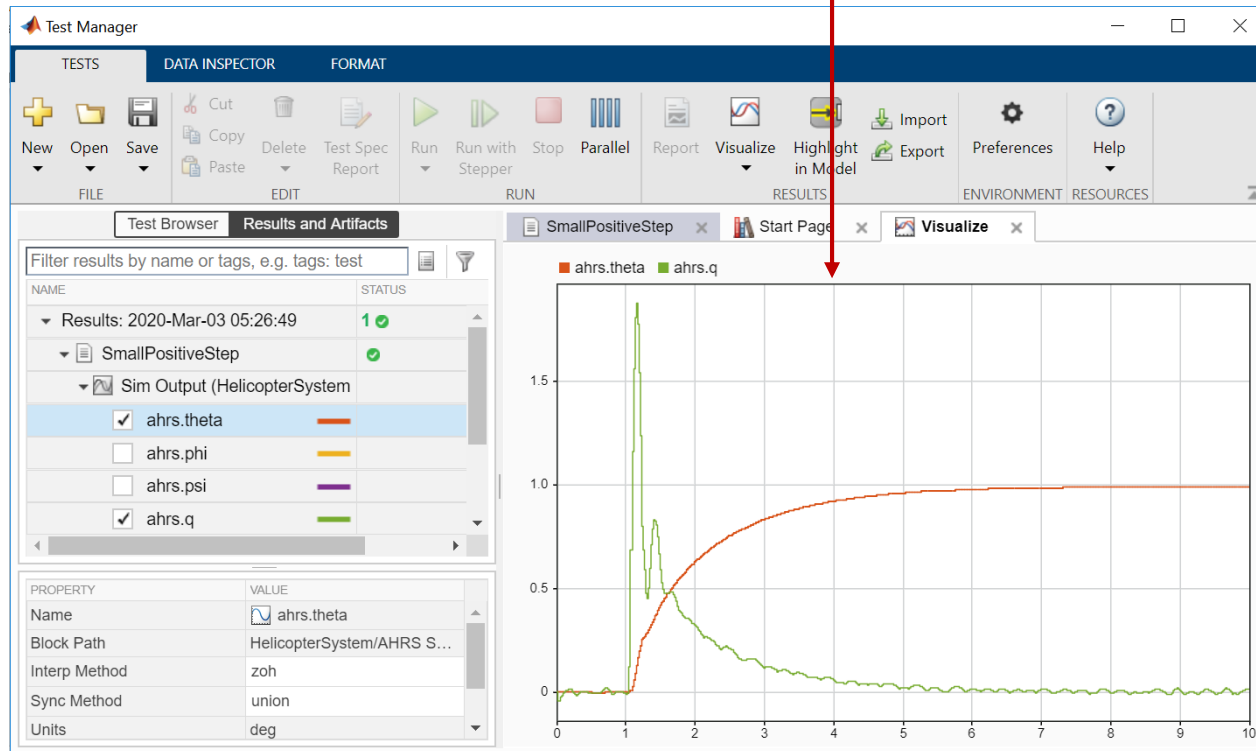
System Level Simulation

Provide Verified Models from Software (DO-178C) and Hardware (DO-254) Processes back to Systems (ARP 4754)

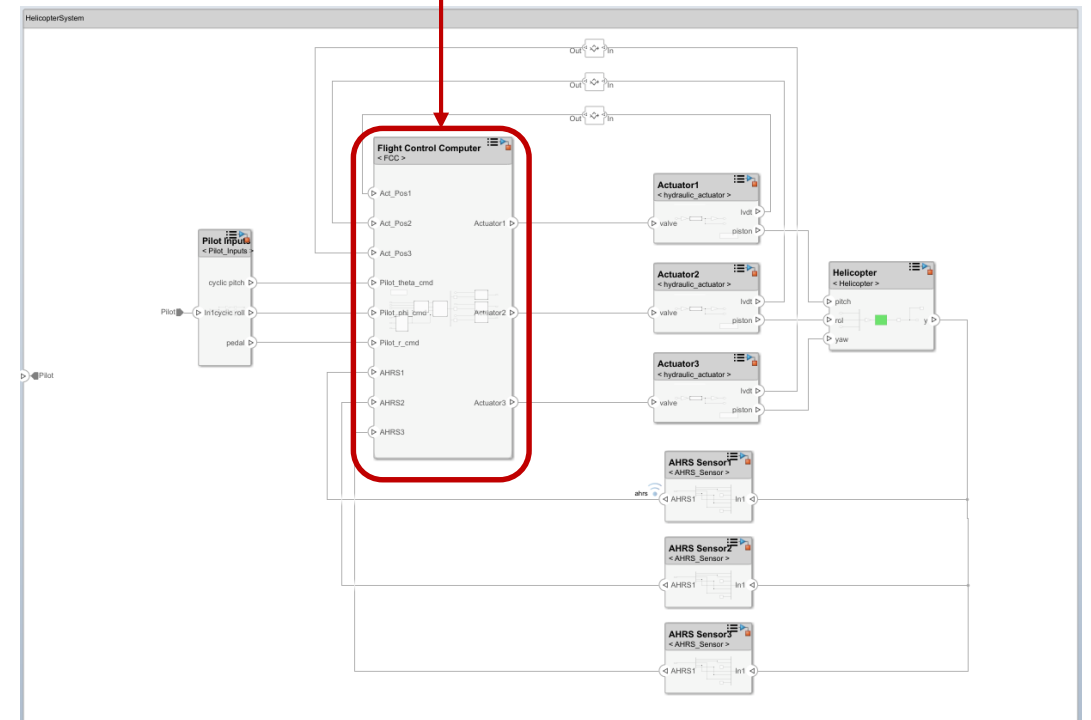


System Simulation using Design Models

System Response



Software and Hardware Design Models

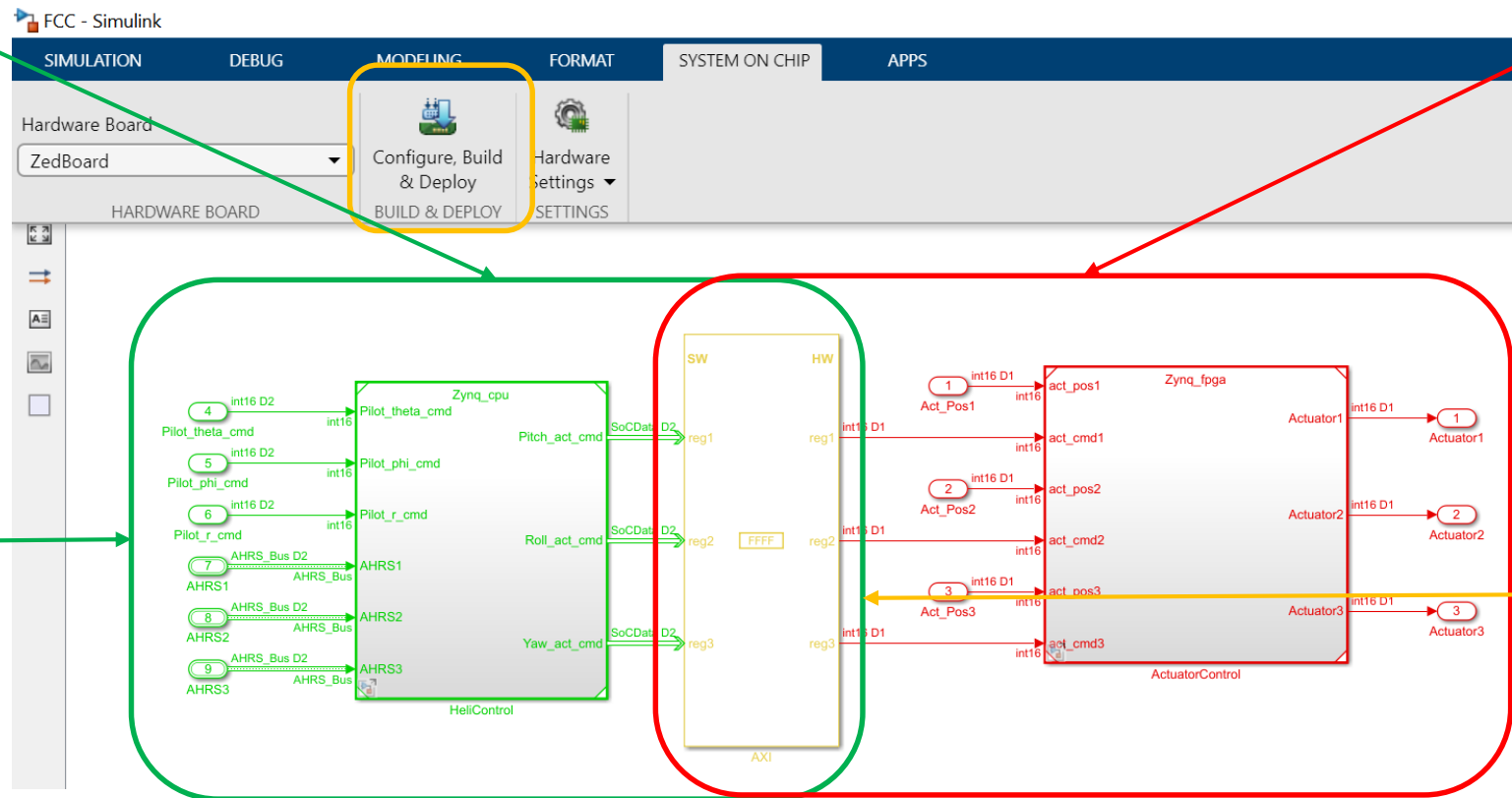


Software and Hardware Implementation

Generate C and VHDL Code and Deploy to SoC

C code

VHDL code



Generate
for multi-core

Glue code
C and VHDL

C Code Generation and Traceability Report

AHRS_voter * - Simulink

SIMULATION DEBUG MODELING FORMAT SYSTEM ON CHIP APPS **C CODE**

Embedded C Code Quick Start C/C++ Code Advisor Settings Code for: AHRS_voter

Generate Code View Code Open Report Remove Highlighting Verify Code Share

OUTPUT ASSISTANCE PREPARE GENERATE CODE RESULTS VERIFY SHARE

Web Browser - AHRS_voter_c.html

AHRS_voter_contents.html AHRS_voter_c.html

Location: EXPO_Project/DO178Software/DO_04_Code/specification/slprj/ert/AHRS_voter/html/AHRS_voter_c.html

```
95 case 3:
96 /* Outputs for IfAction SubSystem: '<Root>/Mid_Value' incorporates:
97  * ActionPort: '<S3>/Action Port'
98  *
99  * Block requirements for '<Root>/Mid_Value':
100  * 1. HLR_11: AHRS Voting for Triple Sensors (HelicopterSoftwareRequirements#15)
101  */
102 /* MinMax: '<S3>/MinMax'
103  *
104  * Block requirements for '<S3>/MinMax':
105  * 1. HLR_11: AHRS Voting for Triple Sensors (HelicopterSoftwareRequirements#15)
106  */
107 if (rtu_AHRS1->theta < rtu_AHRS2->theta) {
108     localB->y = rtu_AHRS1->theta;
109 } else {
110     localB->y = rtu_AHRS2->theta;
111 }
112
113 /* MinMax: '<S3>/MinMax1'
114  *
115  * Block requirements for '<S3>/MinMax1':
116  * 1. HLR_11: AHRS Voting for Triple Sensors (HelicopterSoftwareRequirements#15)
117  */
```

Code Generation Report

Find: Match Case

Contents

- Summary
- Subsystem Report
- Traceability Report**
- Static Code Metrics Report
- Code Replacements Report
- Coder Assumptions

Generated Code

- [-] Model files
 - AHRS_voter.c
 - AHRS_voter.h
- [+] Shared files (1)
- [+] Other files (1)

Traceable Simulink Blocks / Stateflow Objects / MATLAB Functions

Root system: [AHRS_voter](#)

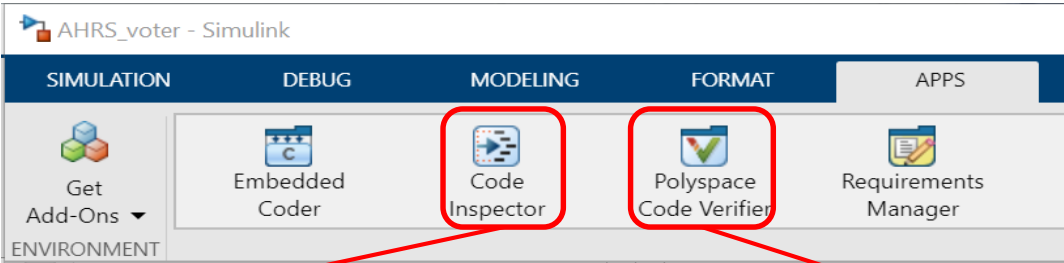
Object Name	Code Location
<Root>/Avg_Value	AHRS_voter.c:391, 394, 481
<Root>/Default	AHRS_voter.c:575, 582
<Root>/Merge	AHRS_voter.c:40, 50, 52, 57, 58, 61 AHRS_voter.h:82, 83
<Root>/Mid_Value	AHRS_voter.c:96, 99, 387
<Root>/Single_Value	AHRS_voter.c:485, 488, 571
<Root>/Sum	AHRS_voter.c:84, 89, 92, 93
<Root>/Switch_Case	AHRS_voter.c:82, 86, 92, 95, 390, 484, 586

Subsystem: [AHRS_voter/Avg_Value](#)

Object Name	Code Location
<S1>/Action Port	AHRS_voter.c:392
<S1>/Constant	AHRS_voter.c:20, 398, 403, 414, 421, 426, 437, 444, 449, 460 AHRS_voter.h:70, 71
<S1>/Gain	AHRS_voter.c:25, 466, 469, 475, 476, 480 AHRS_voter.h:73, 74
<S1>/Sum	AHRS_voter.c:467, 472, 476, 477
<S1>/Switch	AHRS_voter.c:397, 400, 406, 407, 408, 409, 410, 411, 413, 414, 418

Type here to search

C Code Inspection and Formal Analysis



Web Browser - Simulink Code Inspector Report for ActuatorLoop.slx

Simulink Code Inspector Report for ActuatorLoop.slx

Location: /DO_04_Code/verification/code_reviews/ActuatorLoop/ActuatorLoop.slx

Code Verification Results : Verified

Function Interface Verification Results : Verified

Function	Status	Details
ActuatorLoop_Disable	Verified	-
ActuatorLoop_initialize	Verified	-
ActuatorLoop	Verified	-

Model To Code Verification Results : Verified

Status	Count
Model objects with status Verified :	22
Model objects with status Partially processed :	0
Model objects with status Unable to process :	0
Model objects with status Failed to verify :	0

Chapter 1. Polyspace Code Verification Summary

Table 1.1. Code Metrics Summary

Polyspace Code Metrics	Enabled
Pass/Fail	

Table 1.2. Coding Standard Summary - MISRA C:2012 Guidelines

MISRA C:2012 Guidelines	Enabled
Violations	1
Pass/Fail	

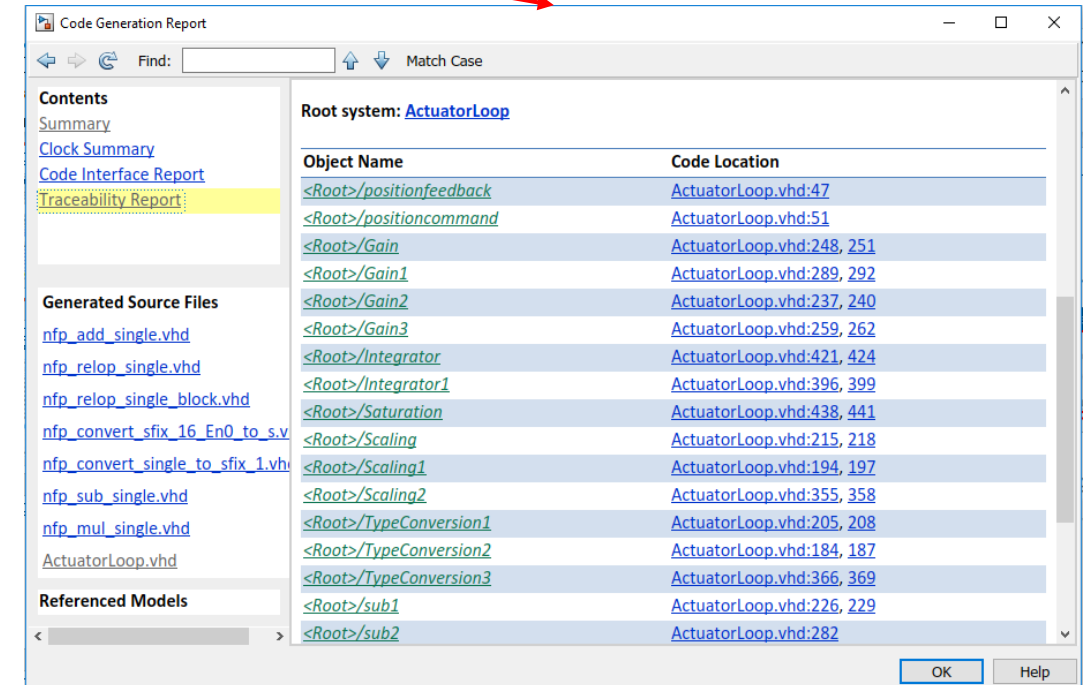
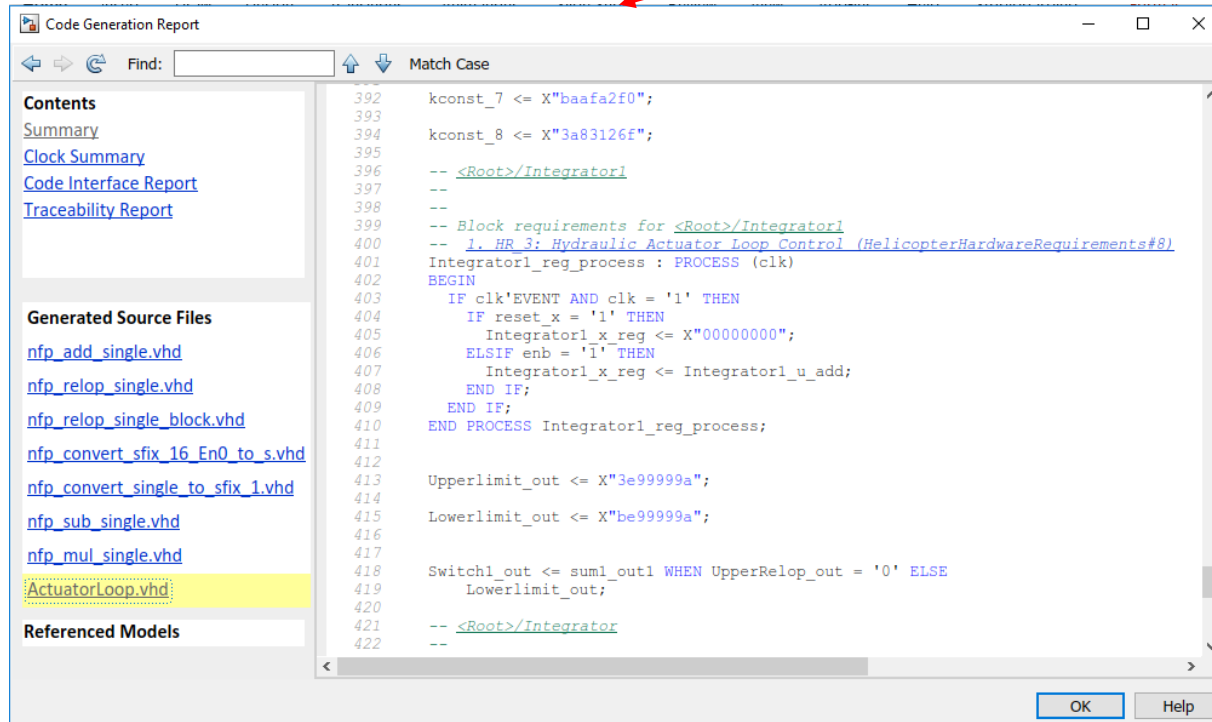
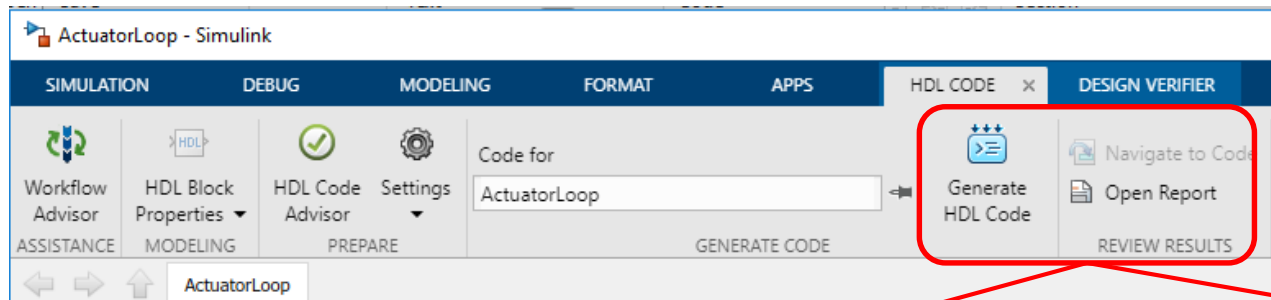
Table 1.3. Run-Time Checks Summary

Run-Time Checks	Enabled
Number of Red Checks	0
Number of Gray Checks	0
Number of Orange Checks	0
Number of Green Checks	706
Proven	100.0%
Pass/Fail	

Table 1.4. Global Variable Summary

Category	Total
Used non-shared variable	11

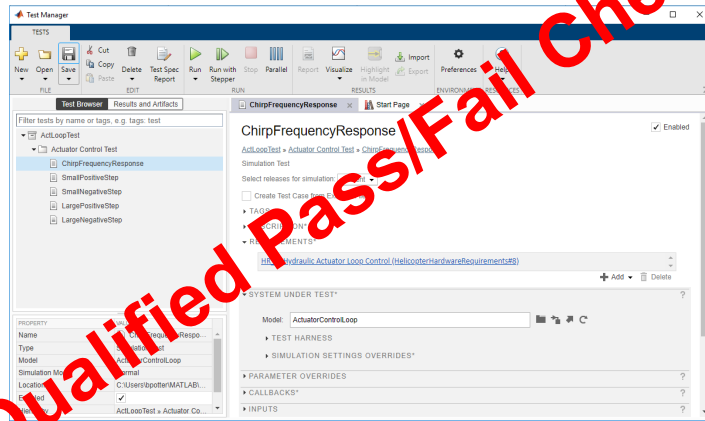
VHDL Code Generation and Traceability Report



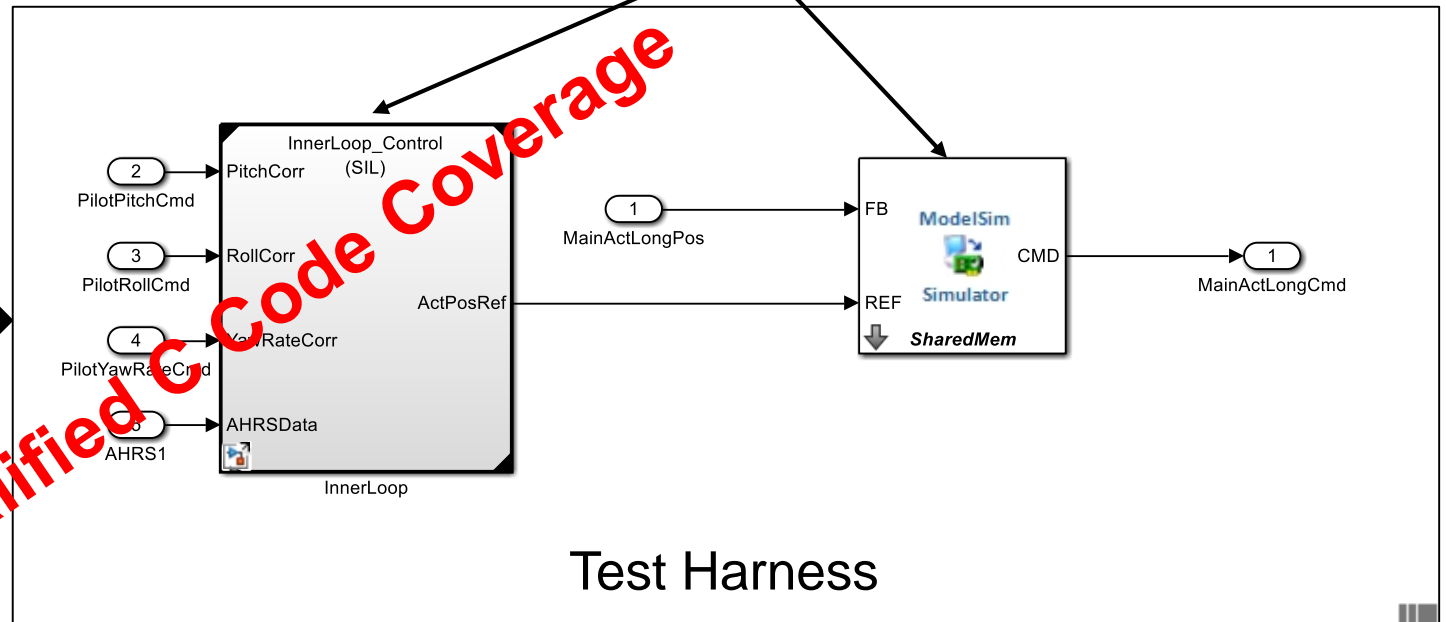
Hardware and Software Testing

Host-Based Software In-The-Loop and Co-Simulation Testing

Test Manager

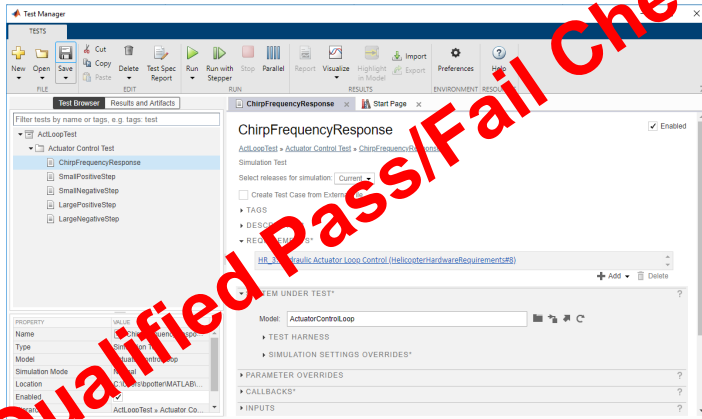


Code Coverage Analysis

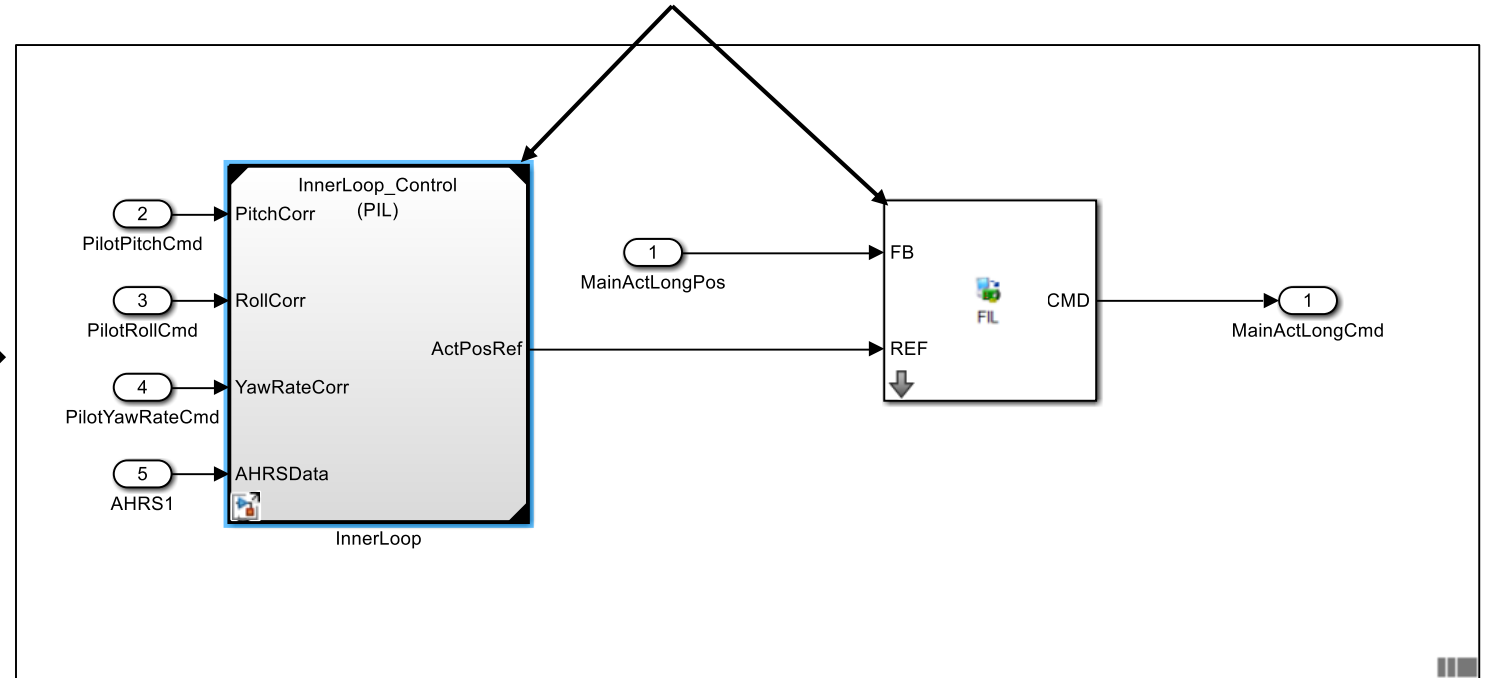


Target-Based Processor and FPGA In-The-Loop Testing

Test Manager

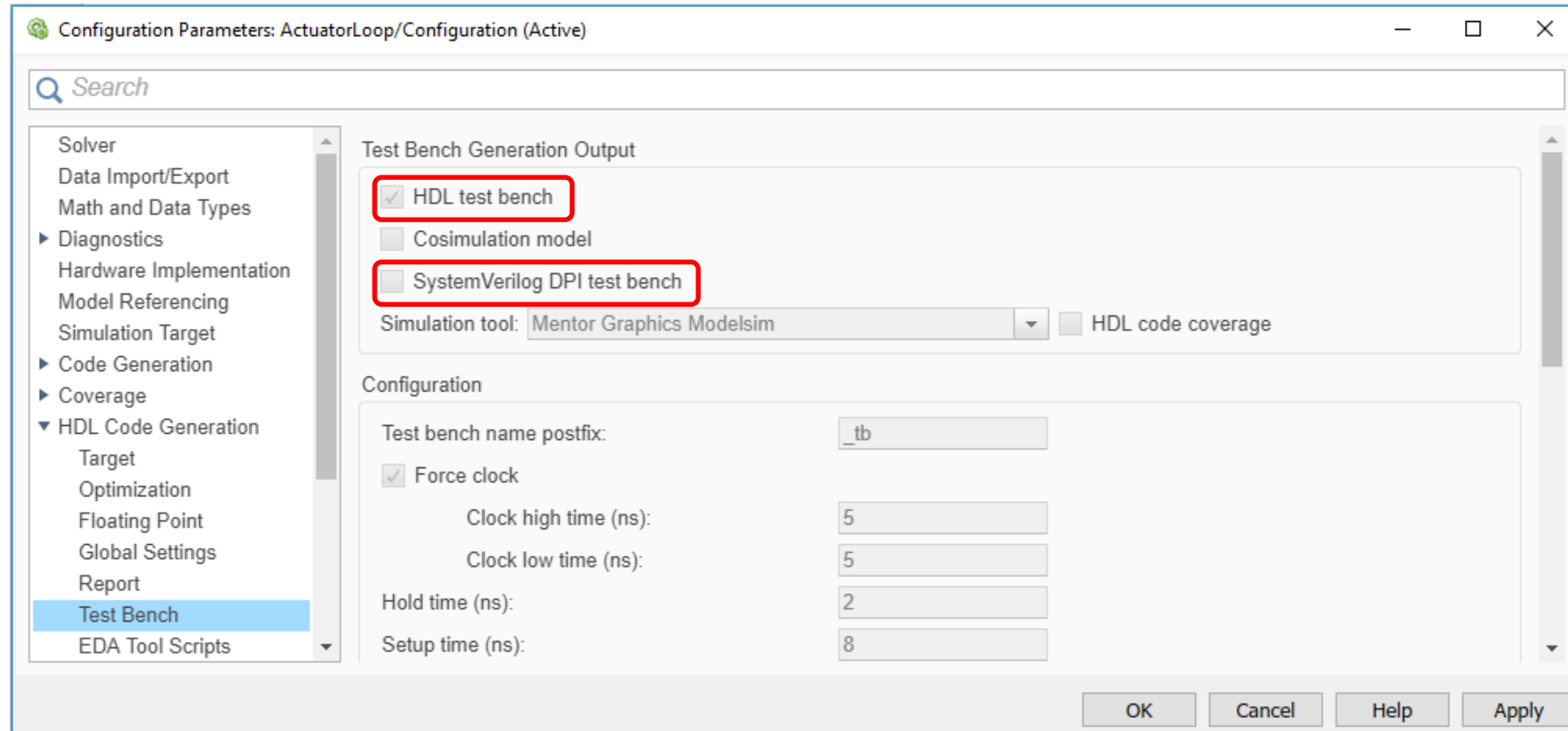


Hardware-Software Integration Tests



Alternate Hardware Testing Methods Using Test Benches

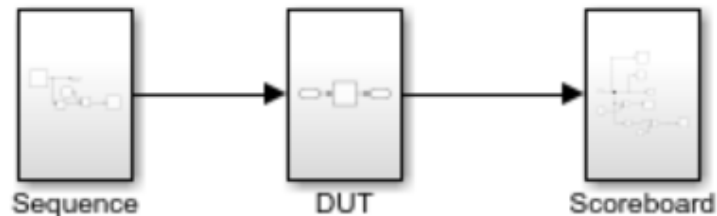
- Simulation cases and Simulink Design Verifier cases are exported to Test Benches



Alternate Hardware Testing Methods using Universal Verification Methodology (UVM)

- Simulation cases and Simulink Design Verifier cases are exported to UVM

```
open_system('hdlv_uvmbuild');
```



Generate UVM Test Bench

Generate a UVM test bench from this Simulink model, specifying the paths to the DUT, sequence, and scoreboard subsystems.

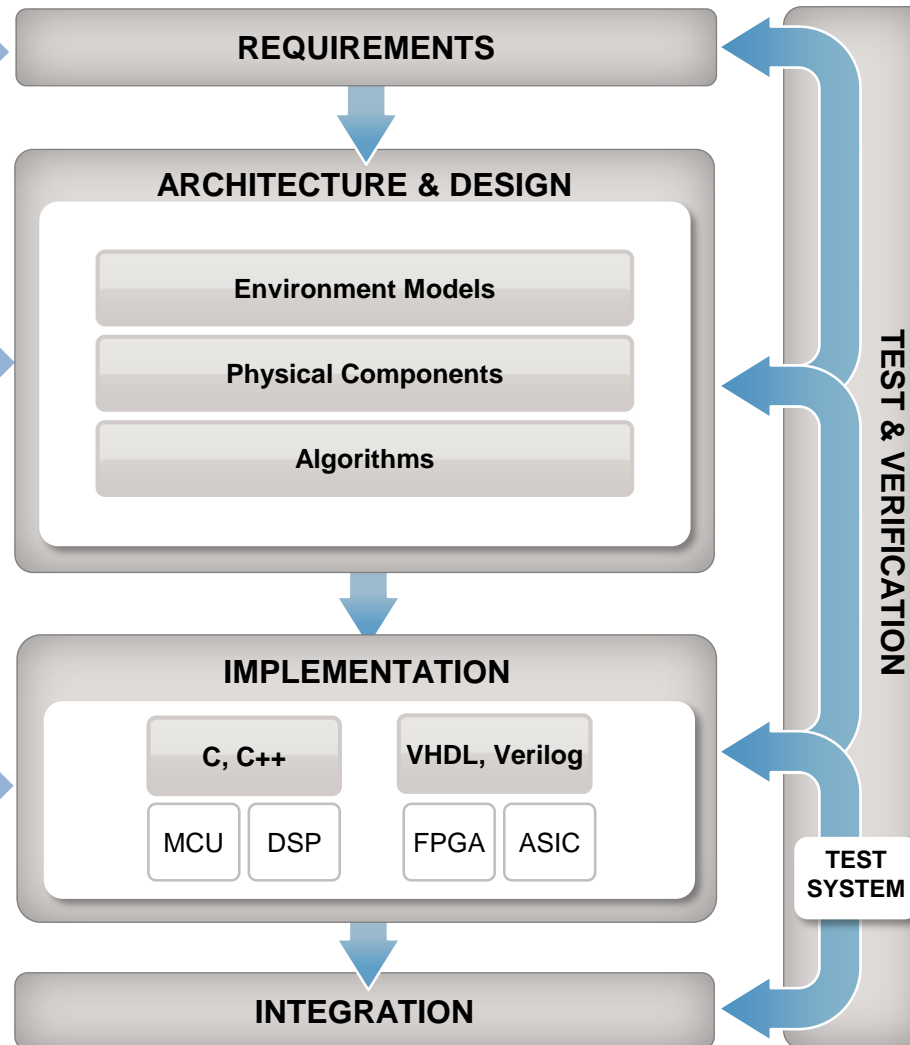
```
uvmbuild('hdlv_uvmbuild/DUT','hdlv_uvmbuild/Sequence','hdlv_uvmbuild/Scoreboard');
```

Qualified Tools ✓

- Simulink ✓
Requirements

- System Composer
- Simulink
- Stateflow

- Embedded Coder
- HDL Coder



- Simulink Check ✓
- Simulink Test ✓
- Simulink Coverage ✓
- Simulink Design Verifier ✓
- Simulink Report Generator ✓

- Simulink Code Inspector ✓
- Polyspace Bug Finder ✓
- Polyspace Code Prover ✓
- HDL Verifier

User Stories and Examples

DO Qualification Kit (for DO-178) — Examples

Tool Qualification

DO-178C Simulink Project Template



Helicopter Flight Control: A Model-Based Design Example for DO-178C an...

Demonstrates the use of the DO-178C project template in a helicopter flight control project.

Airbus Develops Fuel Management System for the A380 Using Model-Based Design



Airbus A380, the world's largest commercial aircraft.

Challenge

Develop a controller for the Airbus A380 fuel management system

Solution

Use MATLAB, Simulink, and Stateflow for Model-Based Design to model and simulate the control logic, communicate the functional specification, and accelerate the development of simulators

Results

- Months of development time eliminated
- Models reused throughout development
- Additional complexity handled without staff increases

"The Simulink and Stateflow models enabled us to validate requirements early and communicate the functional specification to our suppliers, complementing the written requirements in conformance with ARP 4754."

Christopher Slack
Airbus

BAE Systems Delivers DO-178B Level A Flight Software on Schedule with Model-Based Design



Primary flight control computers from BAE Systems.

Challenge

Develop flight-critical software for a mid-sized business jet in compliance with DO-178B Level A standards

Solution

Use Model-Based Design to model the software and systems, run simulations with customer-provided test vectors, trace requirements to model elements, and generate 200,000 lines of certified code

Results

- Development efficiency doubled
- Certification schedule maintained
- Communication between teams facilitated

"When we generated code from our Simulink models with Embedded Coder, the team we handed it off to knew it was gold—that it was debugged and fully met the requirements—because we had run it through the Simulink test vectors supplied by our customer. That was a huge advantage on this program."

Maria Radecki
BAE Systems

Conclusion

- Model-Based Design increases productivity for development of certified systems
- A single development environment for System on a Chip
- Qualified tools that span ARP 4754A, DO-178C and DO-254
- Whether you are a systems engineer, software engineer or hardware engineer, you can deploy Model-Based Design on your certification project

Learn More

- ARP 4754 Solutions Page
 - <https://www.mathworks.com/solutions/aerospace-defense/standards/arp-4754.html>
- DO-178 Solutions Page
 - <https://www.mathworks.com/solutions/aerospace-defense/standards/do-178.html>
- DO-254 Solutions Page
 - <https://www.mathworks.com/solutions/aerospace-defense/standards/do-254.html>
- DO Qualification Kit
 - <https://www.mathworks.com/products/do-178.html>