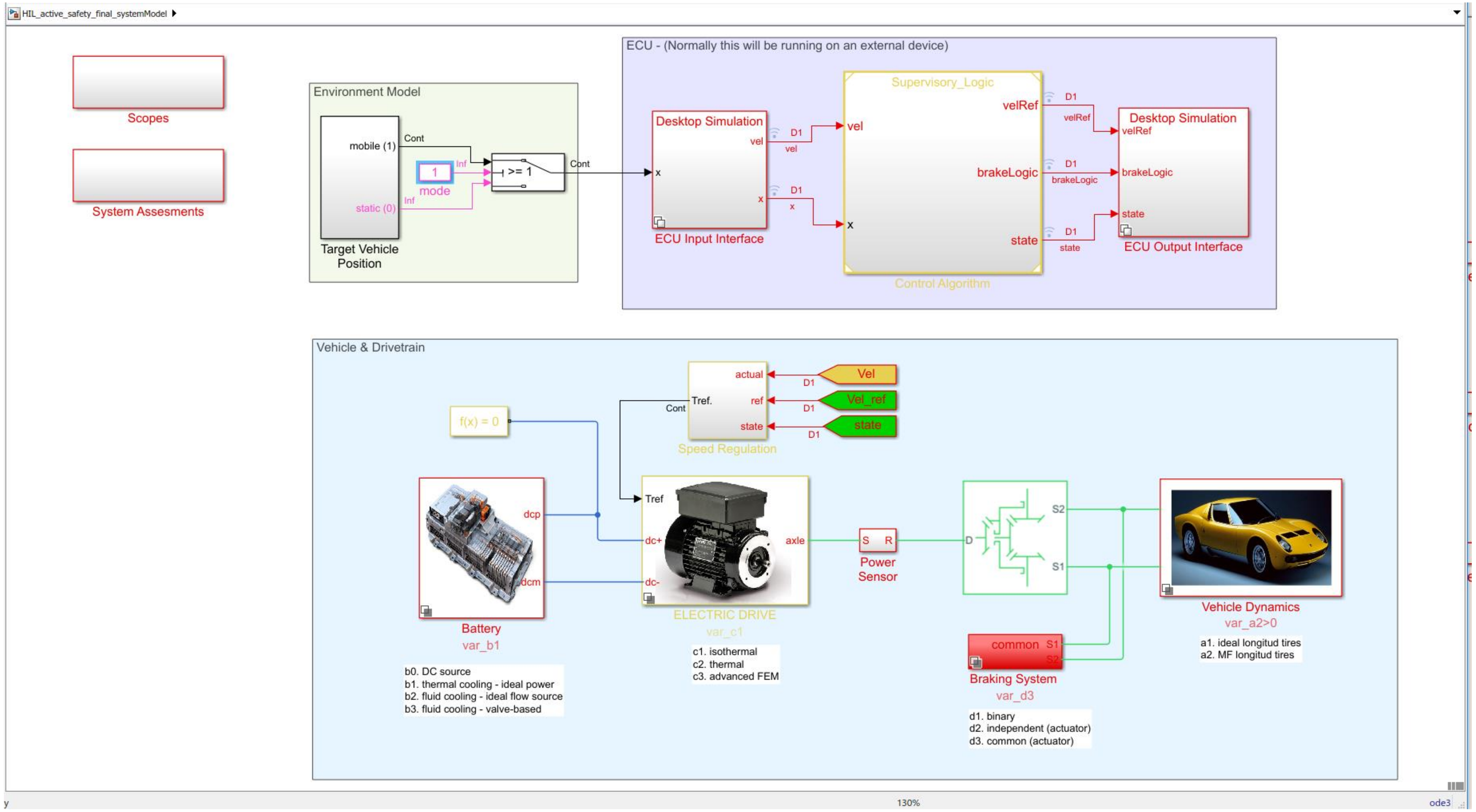
MATLAB EXPO 2018

Real-Time Testing in a Modern, Agile Development Workflow

Simon Eriksson – Application Engineer

Demo – Going from Desktop Testing to Real-Time Testing



b0. DC source
b1. thermal cooling - ideal power
b2. fluid cooling - ideal flow source
b3. fluid cooling - valve-based

c1. isothermal
c2. thermal
c3. advanced FEM

d1. binary
d2. independent (actuator)
d3. common (actuator)

a1. ideal longitud tires
a2. MF longitud tires

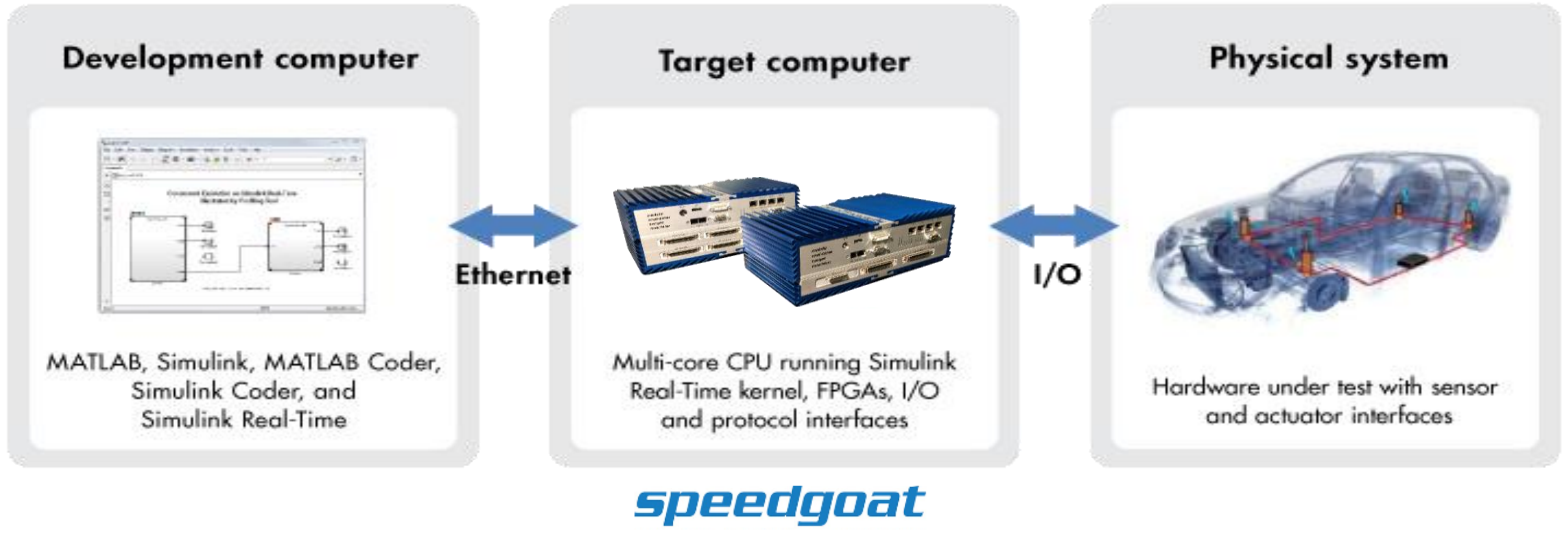
Key Take-Aways From This Presentation

- Agile = iterative and short design cycles, high degree of re-use of digital assets is essential

- Technology landscape demands faster control loops, prototyping with SW is not enough anymore

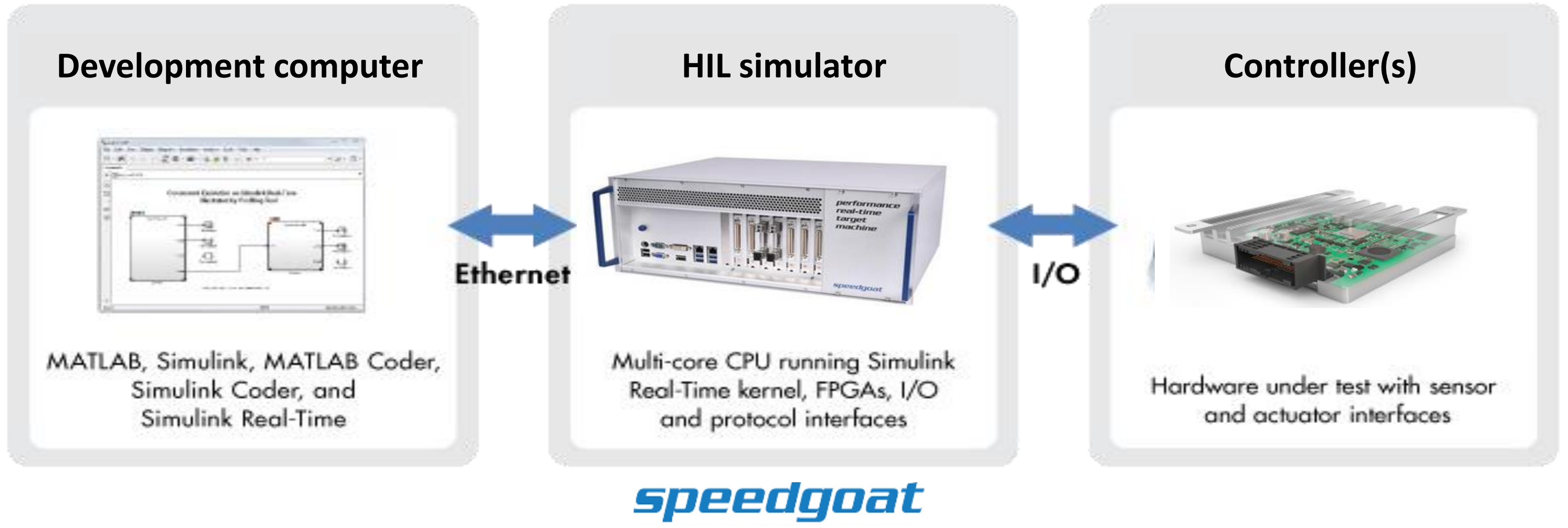
Overview on Real-Time Testing

- Rapid Control Prototyping
  - Run your algorithm against real HW without any manual coding
  - Tune your algorithm live directly from Simulink


Development computer
Target computer
Physical system
Ethernet
I/O
MATLAB, Simulink, MATLAB Coder, Simulink Coder, and Simulink Real-Time
Multi-core CPU running Simulink Real-Time kernel, FPGAs, I/O and protocol interfaces
Hardware under test with sensor and actuator interfaces
speedgoat

Overview on Real-Time Testing

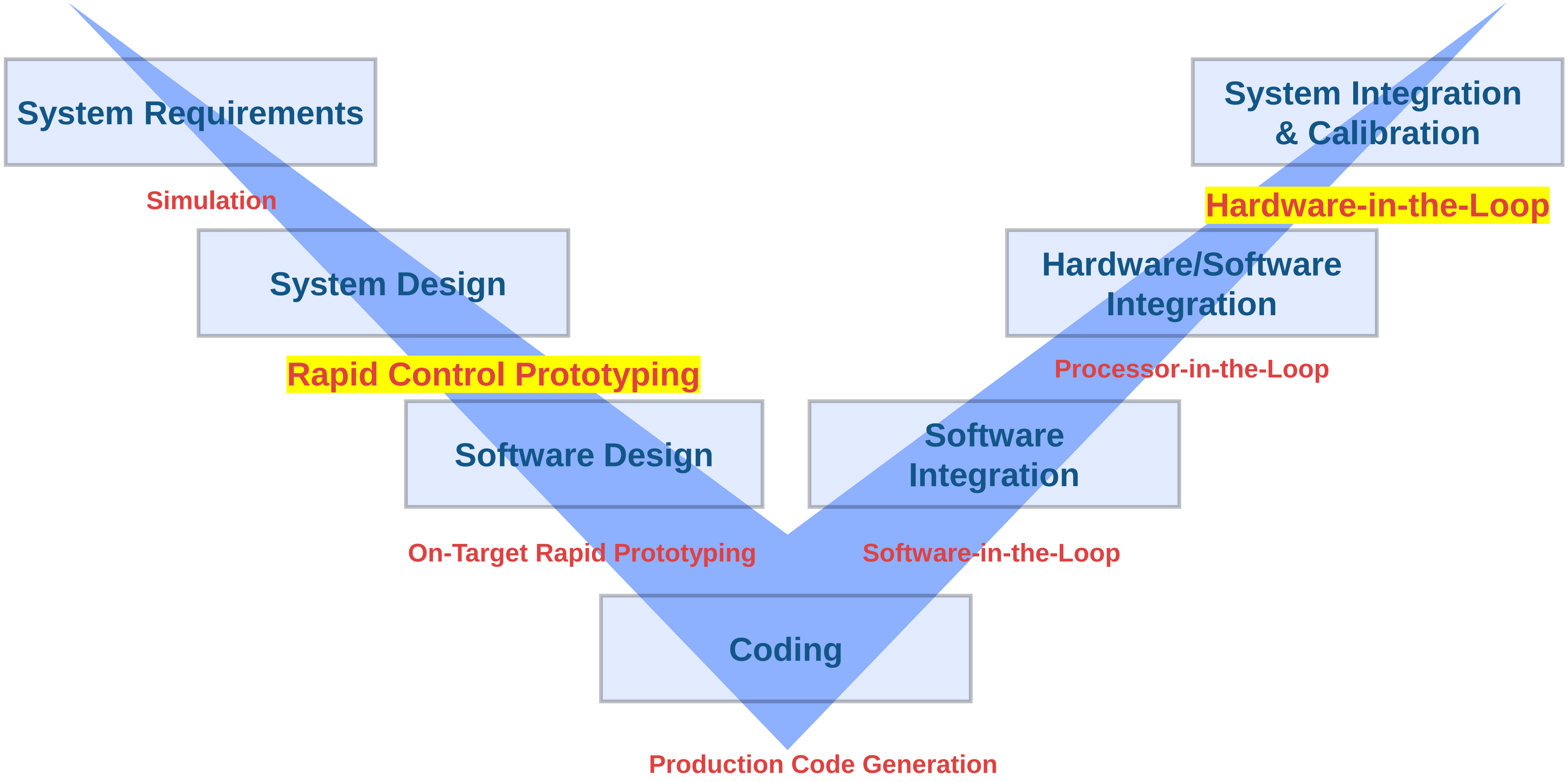
- Hardware-In-the-Loop (HIL)
  - Test your controls against a virtual plant model
  - Test with real I/O, without breaking anything!



Development computer
HIL simulator
Controller(s)
Ethernet
I/O
MATLAB, Simulink, MATLAB Coder, Simulink Coder, and Simulink Real-Time
Multi-core CPU running Simulink Real-Time kernel, FPGAs, I/O and protocol interfaces
Hardware under test with sensor and actuator interfaces
speedgoat

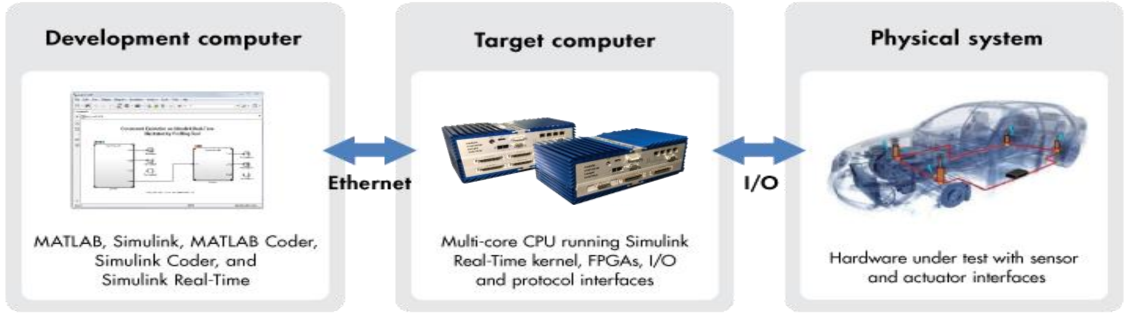
Real-Time Simulation and Testing

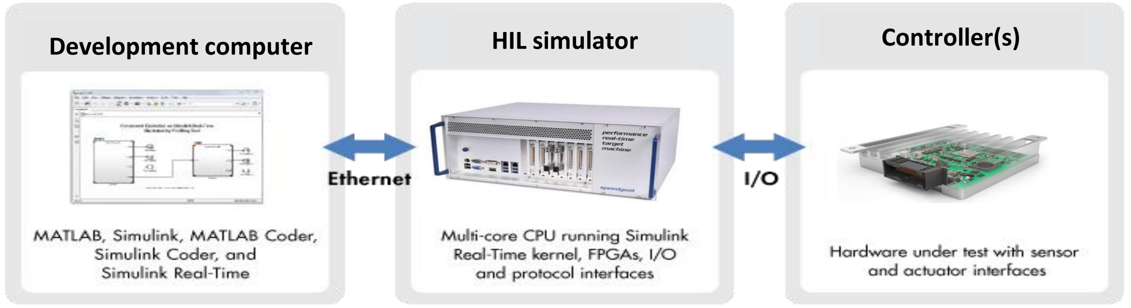
Simulink Real-Time for RCP and HIL



Agile and Real-Time Testing

- Rapid Control Prototyping
  - Get ideas in front of stakeholders
    - Changes in SW
    - Changes in HW
    - Deployable prototypes

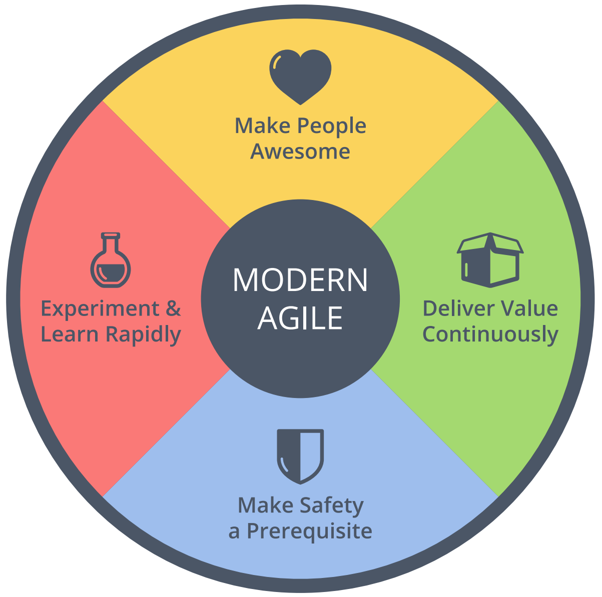

Development computer
MATLAB, Simulink, MATLAB Coder, Simulink Coder, and Simulink Real-Time
Ethernet
Target computer
Multi-core CPU running Simulink Real-Time kernel, FPGAs, I/O and protocol interfaces
I/O
Physical system
Hardware under test with sensor and actuator interfaces

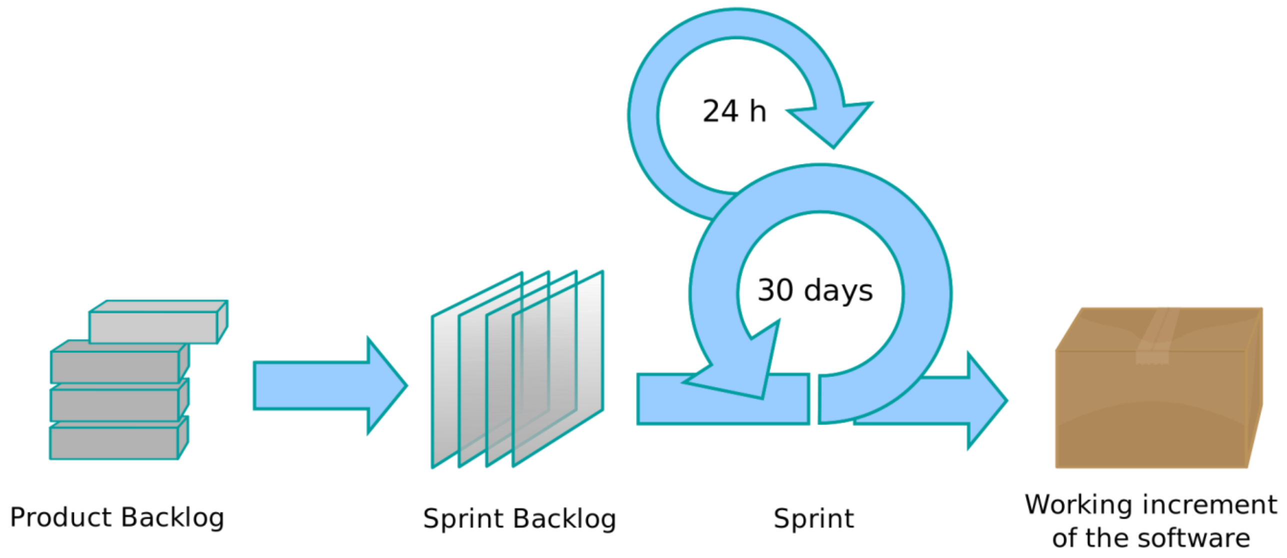

Development computer
MATLAB, Simulink, MATLAB Coder, Simulink Coder, and Simulink Real-Time
Ethernet
HIL simulator
Multi-core CPU running Simulink Real-Time kernel, FPGAs, I/O and protocol interfaces
I/O
Controller(s)
Hardware under test with sensor and actuator interfaces

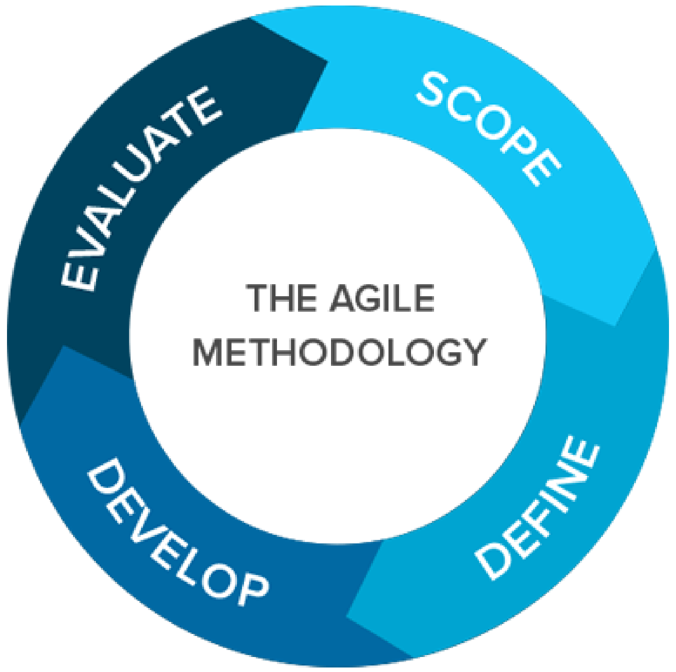
- HIL
  - CI to include HIL as well
    - New/changing requirements
    - “back-to-back” MIL, SIL, PIL and HIL testing

How To Stay Fast/Effective i.e. Agile?

Agile = iterative and with "short" design cycles


Make People Awesome
Experiment & Learn Rapidly
MODERN AGILE
Deliver Value Continuously
Make Safety a Prerequisite

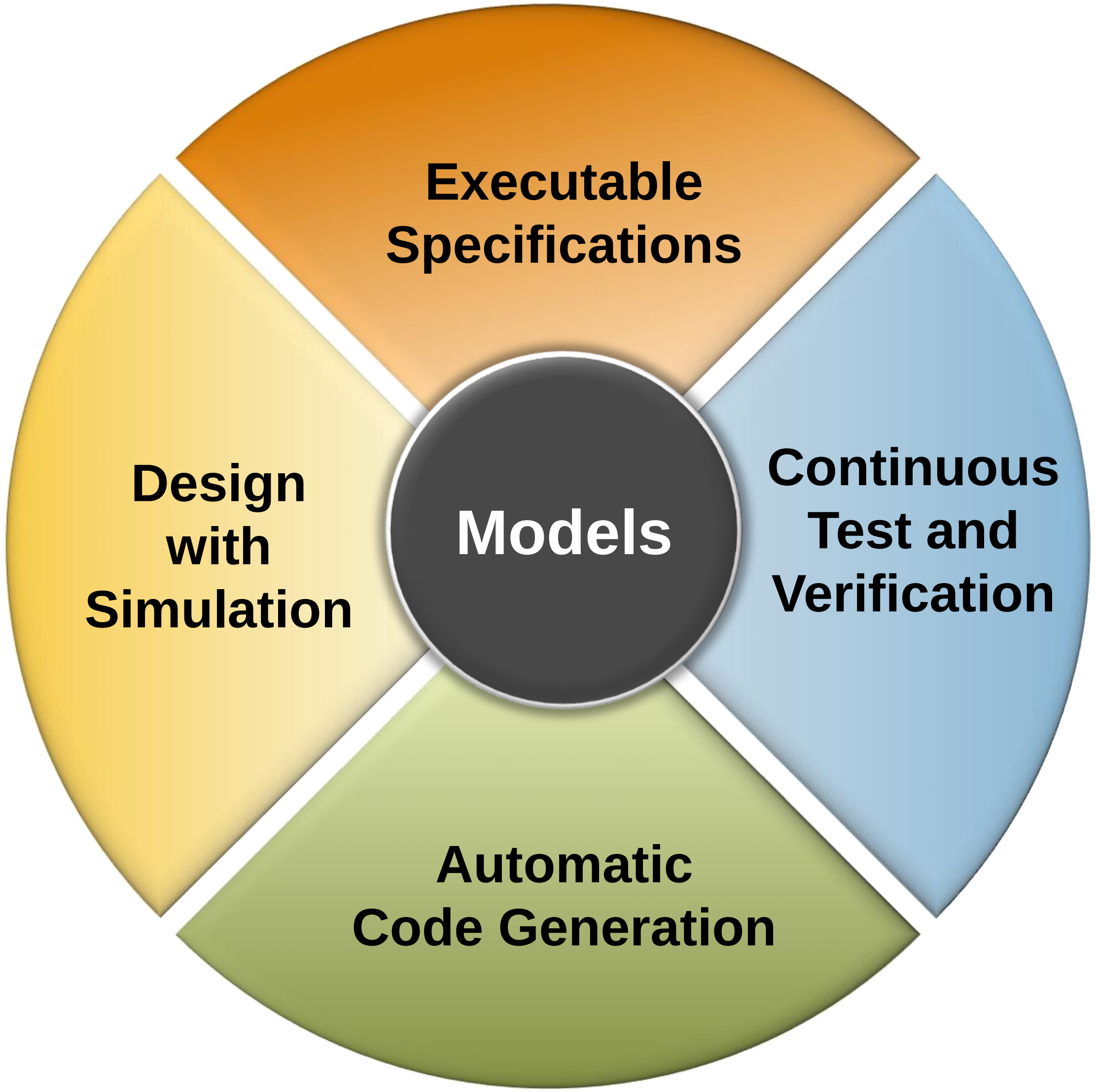




http://www.yodiz.com/
http://modernagile.org/
http://alphatechglobal.com/

How To Stay Fast/Effective i.e. Agile?

- MBD → *Re-use* of digital assets!
- Digital Assets :
  - Models
  - Test Vectors
  - Field Data
  - Requirements
  - Legacy code
  - …


Executable Specifications
Design with Simulation
Models
Continuous Test and Verification
Automatic Code Generation

Rapid Controls Prototyping

Rapid Controls Prototyping - Challenges and Solutions

"Push button" workflow for both SW and HW

- Challenges
  - Faster control loops. Before kHz range now MHz
  - Many teams/competencies (SW and HW) required

- Solutions
  - Improved HDL code generation allows for prototyping on FPGA
  - Integration with Simulink Real-Time enables connectivity to wide range of I/O *combined* with FPGAs without manual coding

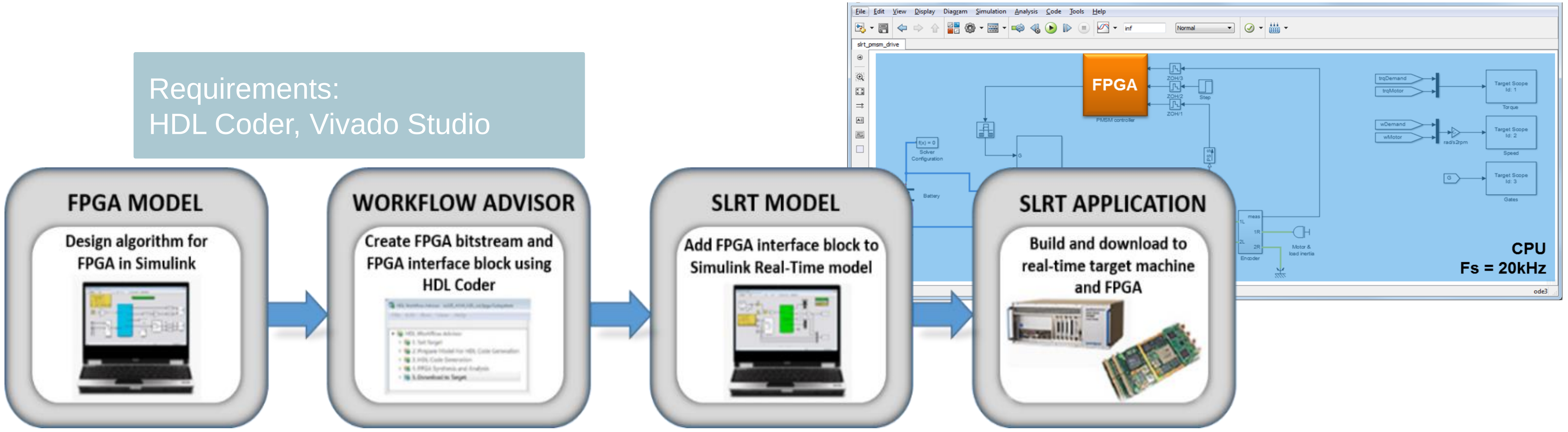
Rapid Controls Prototyping
On HW

Create FPGA I/O and Algorithmic Subsystems

Accelerate parts of your Simulink model using automated HDL code generation

- Achieve closed-loop sample rates up to several MHz
- Quick reconfiguration of FPGA I/O promotes a flexible real-time testing environment and very fast design iterations

Requirements:
HDL Coder, Vivado Studio

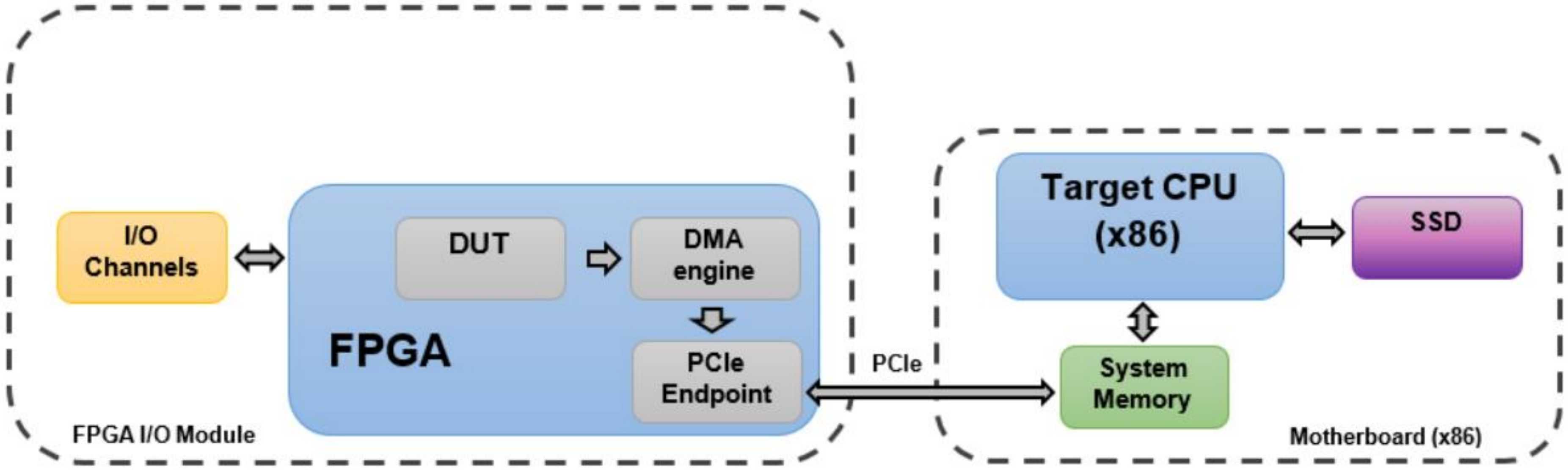

FPGA MODEL
Design algorithm for FPGA in Simulink
WORKFLOW ADVISOR
Create FPGA bitstream and FPGA interface block using HDL Coder
SLRT MODEL
Add FPGA interface block to Simulink Real-Time model
SLRT APPLICATION
Build and download to real-time target machine and FPGA
FPGA
PMSM controller
CPU
Fs = 20kHz

Simulink Programmable FPGA I/O modules

CPU to FPGA communication – DMA

Usage:

- High data throughput between FPGA and x86 CPU
- Low latency link between FPGA and x86 CPU



Simulink Programmable FPGA I/O modules

CPU to FPGA communication – DMA

Use Cases:

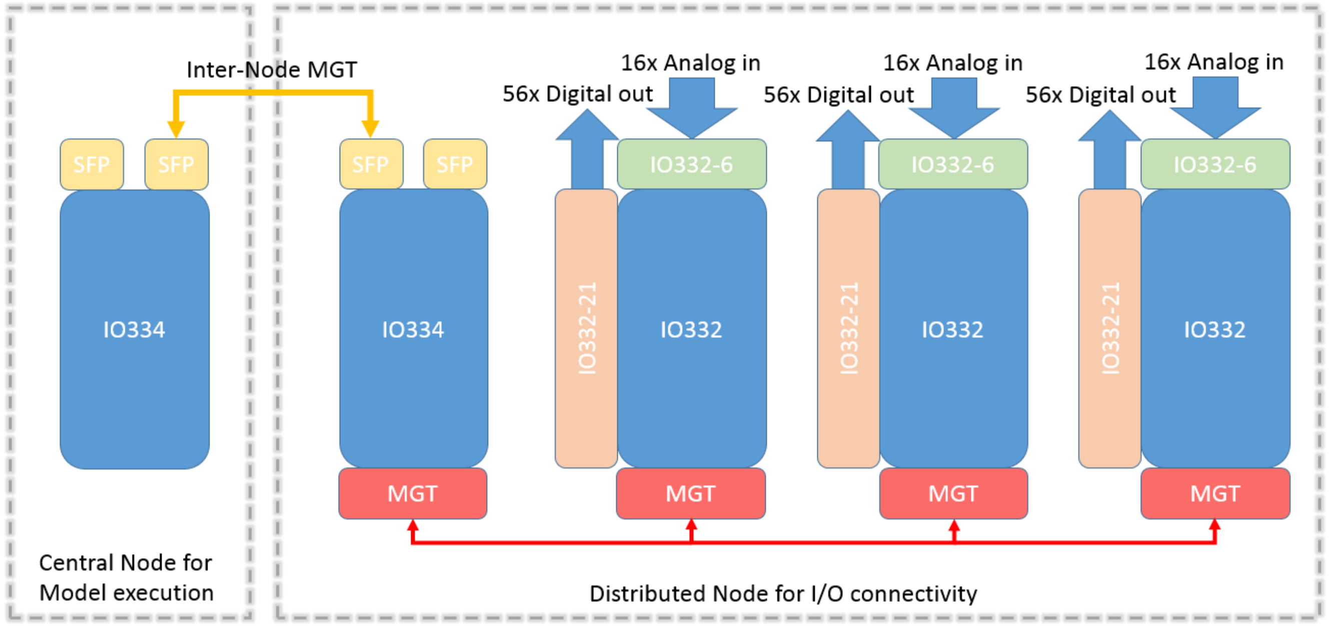
- Direct Streaming
  - High Speed Data logging (FPGA to CPU)
  - Data playback (CPU to FPGA)

- Onboard RAM
  - Failure debugging with Ring Buffer

- Co-processor modes
  - Interrupt based
  - Polling mode based

FPGA-based I/O Modules

Multiple Interconnected FPGA-based I/O Modules



Electric motors (HIL)

Solar inverters for MW grids

100kW DC-DC converter

- Closed-loop rates of 500 kHz and faster
- > 100 analog and digital I/O lines
- Low latency FPGA interconnects (SPI and Aurora protocols)
- Algorithmic execution over multiple FPGAs

Native Floating-Point

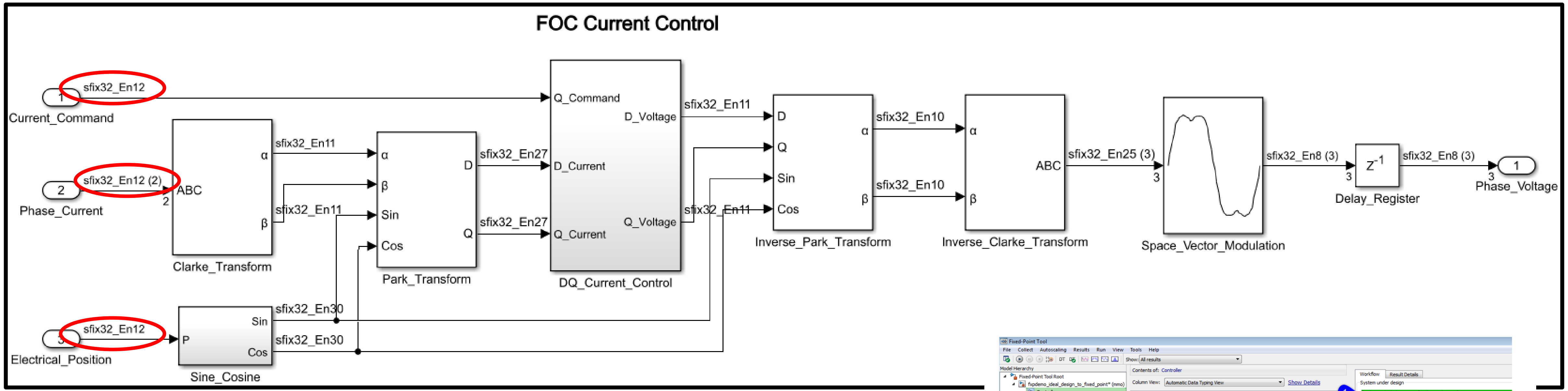
Native floating-point support in HDL Coder for Real-Time Testing

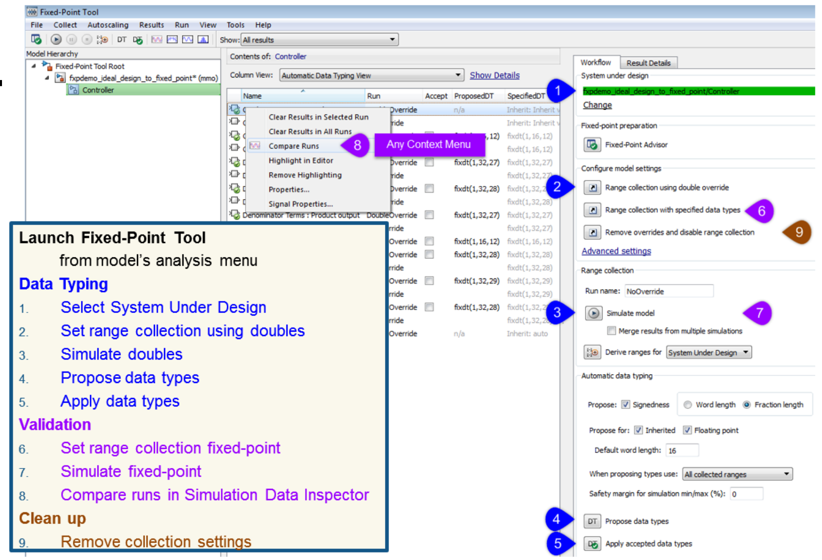
Ok cool, but what does this mean for *me*?

- Rapid Control Prototyping
  - Less effort going to FPGA. ← i.e. more people can do it and be faster too!
  - Create golden reference. ← "this is how good it can be", prove design decisions.

- HIL Testing
  - Easier to take models of physical systems, which by nature can be numerically sensitive, to run on FPGA ← Less risk and time spent creating plant models for real-time

Before : Floating- Point Design Needed to be Converted to Fixed-Point to Generate HDL Code





Launch Fixed-Point Tool from model's analysis menu

Data Typing

1. Select System Under Design
2. Set range collection using doubles
3. Simulate doubles
4. Propose data types
5. Apply data types

Validation

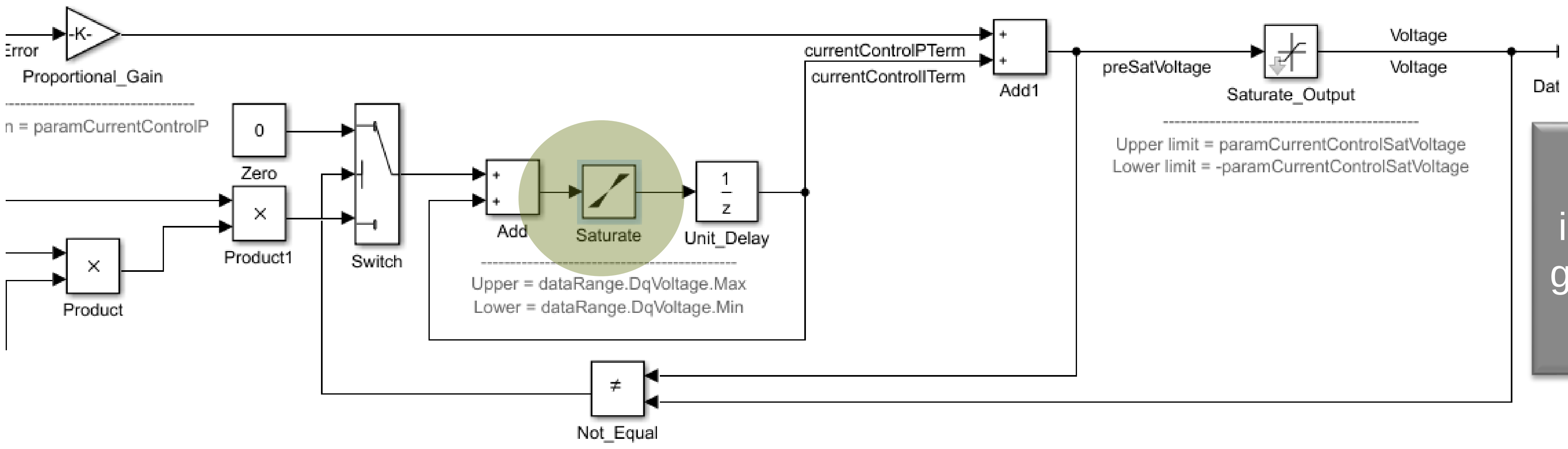
6. Set range collection fixed-point
7. Simulate fixed-point
8. Compare runs in Simulation Data Inspector

Clean up

9. Remove collection settings

Sometimes you need to start and stay in floating point to …

- Implement algorithms with large or unknown dynamic ranges (i.e. integrators in feedback loops)
- Implement operations that are difficult to design in fixed-point (i.e. atan2)

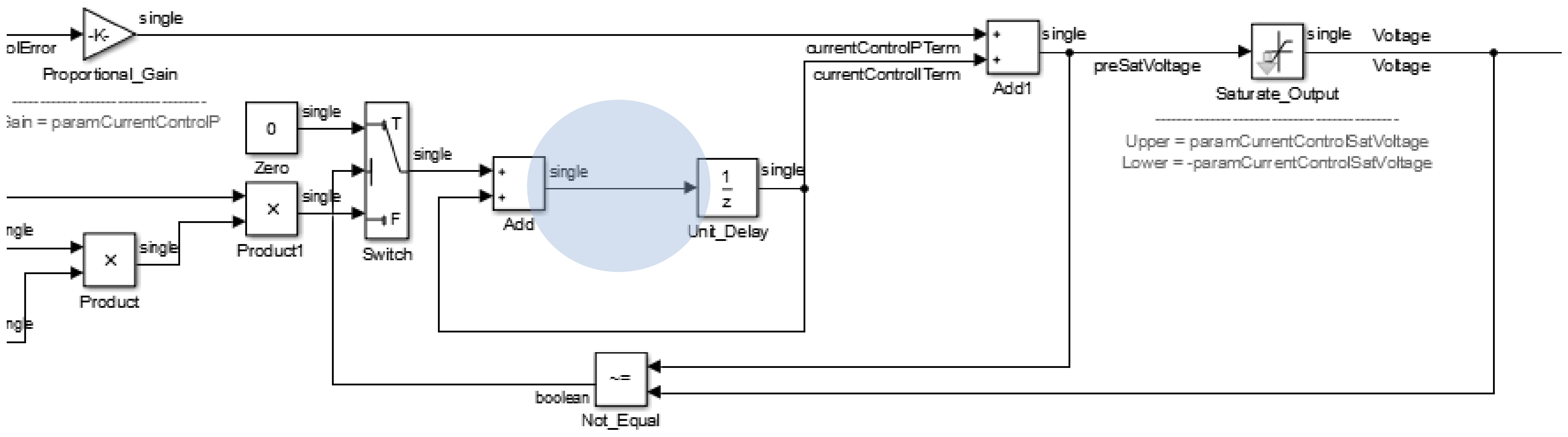


Requires extra integrator saturation to guarantee mathematics will not overflow

Native floating-point support in HDL Coder

- Vendor-independent RTL for FPGA and/or ASIC design
- Full range of IEEE-754 features
  - Optional support for Denormals, INF, NAN, Rounding, …
- Extensive Math and Trigonometry Block support

Operators
Add, Subtract
Mul, Gain, Abs
Min, Max, Relops
Divide, Mod, Rem
Reciprocal, Sin, Cos
Atan, Atan2
Log, Exp, Power
Sqrt & Inverse Sqrt
Data type conversions



Native single-precision support for over 130 Simulink blocks

Hardware-In-the-Loop

Hardware-In-the-Loop and it’s *Agile* Benefits

- HIL enables more efficient/agile testing by
  - Being Repeatable
  - Being Scalable
  - “Failing” without breaking anything
  - Automation

HIL - Challenges and Solutions

Automation and re-use going from testing on desktop and in real-time

- Challenges
  - Sharing test data (test vectors, test results)
  - HW installations can become large or a highly booked resource
  - Numerical sensitivity when creating plant models

- Solutions
  - Integrated desktop and real-time testing framework
  - New high performance formfactors allow for “desktop HIL”
  - Support for native floating-point allow to stay in single precision when running on FPGA

HIL Systems

Typical HW setups

- Intel Core i7 4.2 GHz quad core, or two Xeon CPUs with 20 cores
- Over 100 I/O modules installable leveraging additional expansion chassis

- Challanges :
  - Usually located in large labs, that require timeslots to be booked etc.
  - Not very mobile for use outside the lab

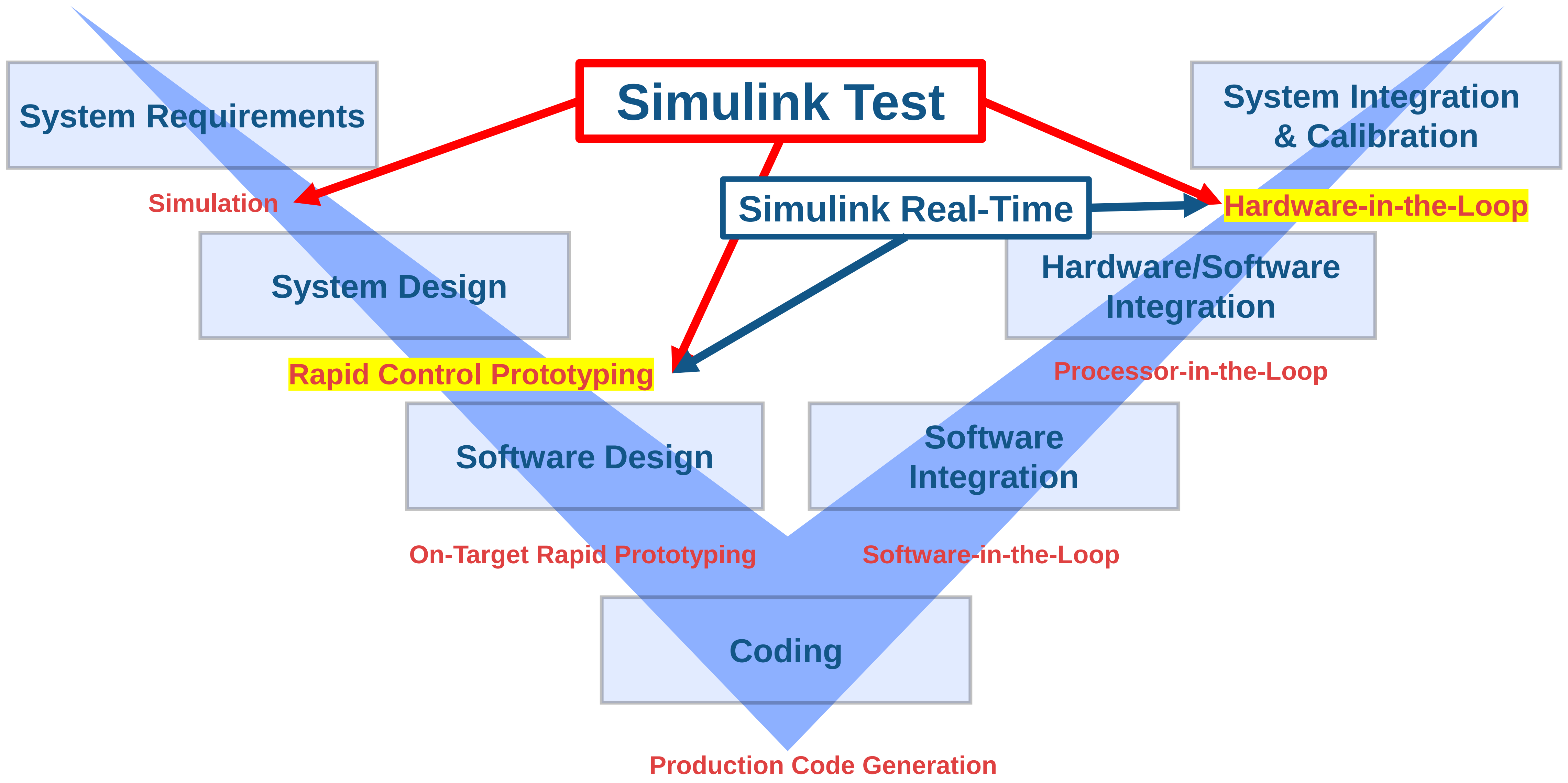
“Desktop HIL”

Power in a Small Form Factor

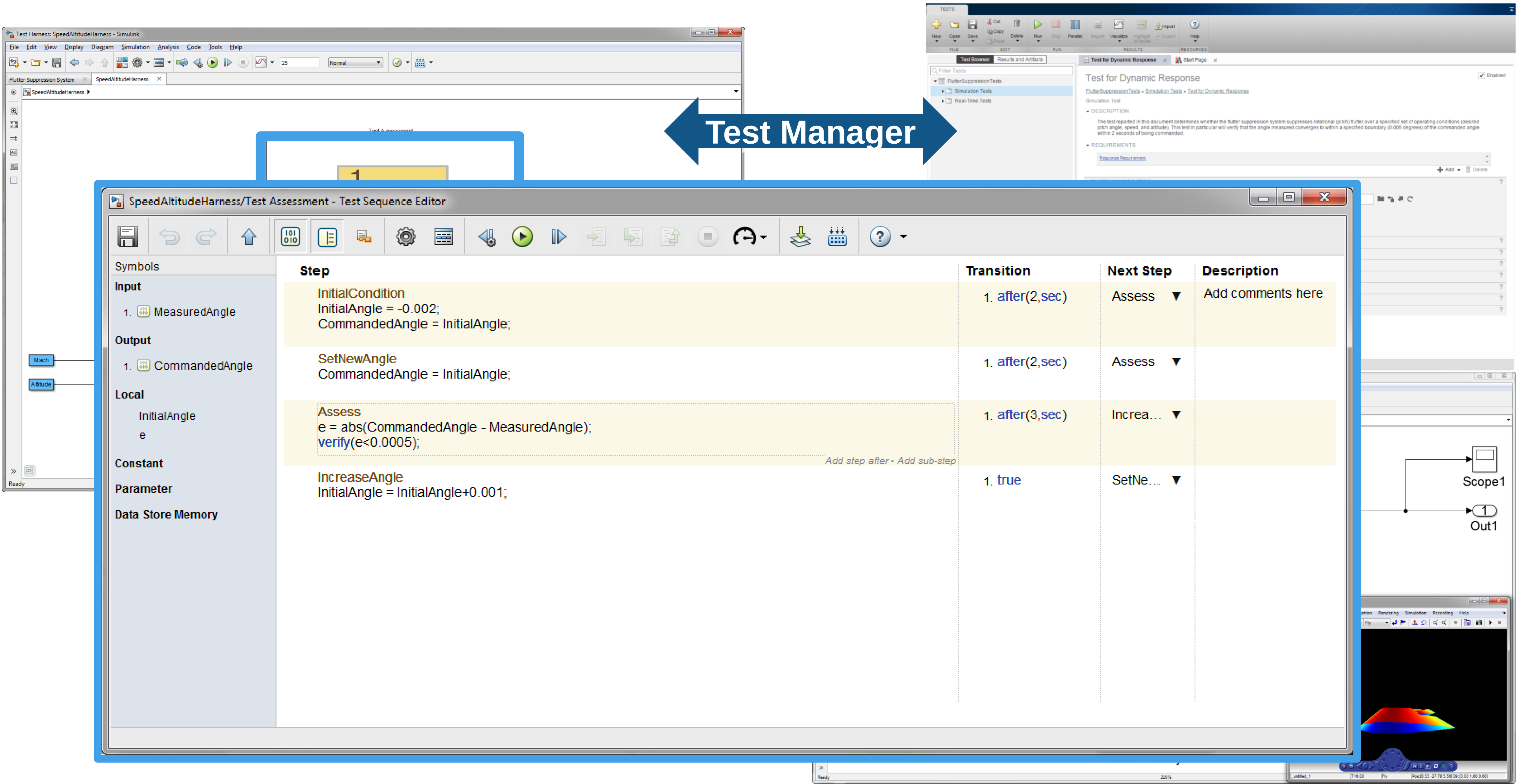
- Solution
  - Ideal for mobile use/“desktop HIL” of for Rapid Control Prototyping
  - Intel Core i7 2.5 GHz dual core CPU and FPGAs
  - Stackable: Up to 14 I/O modules supported
  - **Multi-node HIL-Simulator** for automated testing of large scale plants

(e.g. 64 Profibus / 32 Profinet / Analog I/O / Digital I/O)

Product Development Process



Fully Tested Algorithm in Simulink Test


Test Manager
Test Harness: SpeedAltitudeHarness - Simulink
Test for Dynamic Response
Mach
Altitude
Scope1
Out1
Ready

SpeedAltitudeHarness/Test Assessment - Test Sequence Editor

Symbols
Input
1. MeasuredAngle
Output
1. CommandedAngle
Local
InitialAngle
e
Constant
Parameter
Data Store Memory

Step	Transition	Next Step	Description

InitialCondition
InitialAngle = -0.002;
CommandedAngle = InitialAngle;
	1. after(2,sec)	Assess	Add comments here

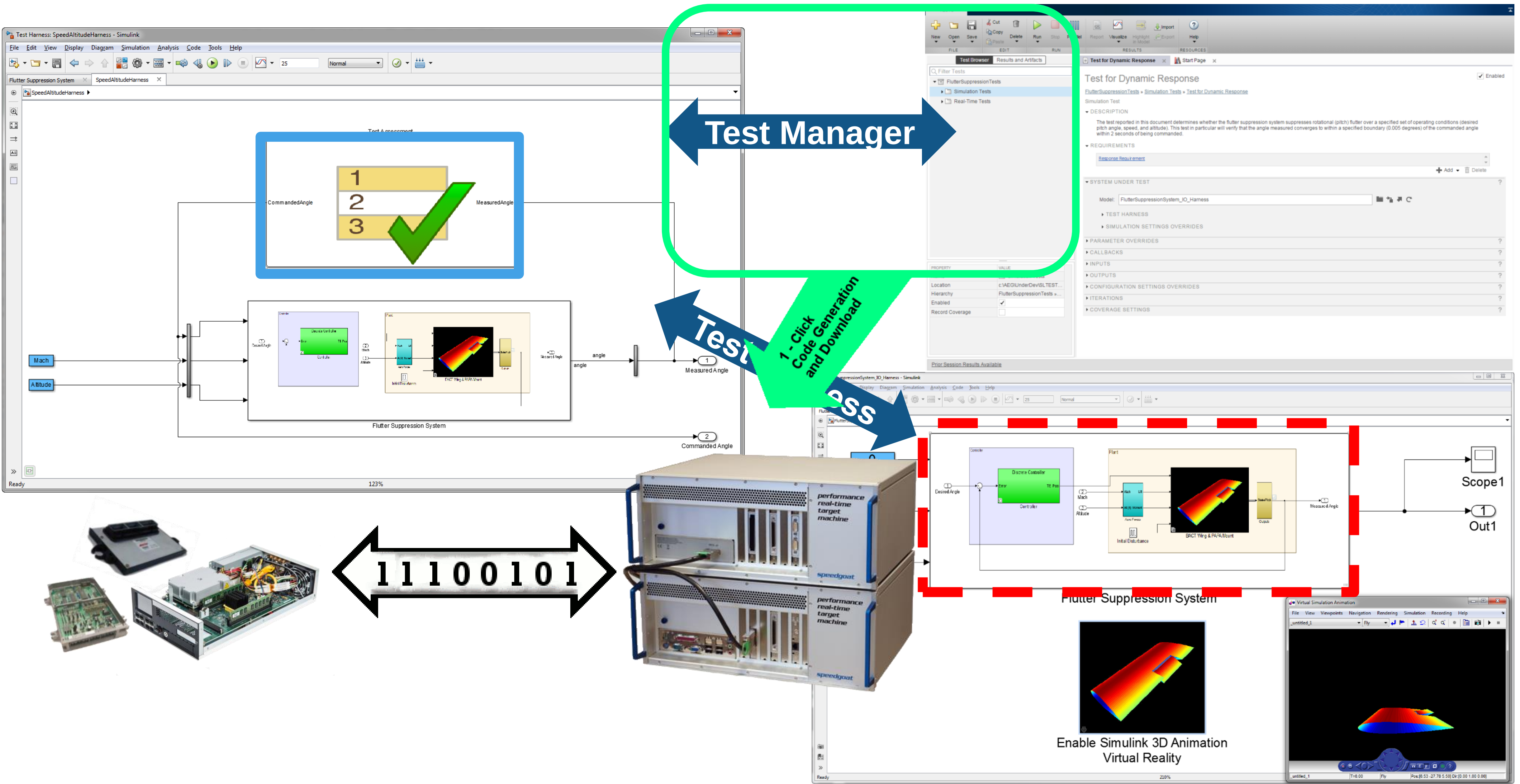
SetNewAngle CommandedAngle = InitialAngle;	1. after(2,sec)	Assess	

Assess
e = abs(CommandedAngle - MeasuredAngle);
verify(e<0.0005);
	1. after(3,sec)	Increa…	

IncreaseAngle InitialAngle = InitialAngle+0.001;	1. true	SetNe…	

Add step after • Add sub-step

Fully Tested Algorithm in Simulink Test


Test Harness: SpeedAltitudeHarness - Simulink
Test Assessment
1
2
3
CommandedAngle
MeasuredAngle
Mach
Altitude
Flutter Suppression System
Measured Angle
Commanded Angle
Test Manager
Test Harness
1 - Click Code Generation and Download
1 1 1 0 0 1 0 1
performance real-time target machine
speedgoat
Scope1
Out1
Flutter Suppression System
Enable Simulink 3D Animation
Virtual Reality

Test for Dynamic Response

FlutterSuppressionTests » Simulation Tests » Test for Dynamic Response

Simulation Test

DESCRIPTION

The test reported in this document determines whether the flutter suppression system suppresses rotational (pitch) flutter over a specified set of operating conditions (desired pitch angle, speed, and altitude). This test in particular will verify that the angle measured converges to within a specified boundary (0.005 degrees) of the commanded angle within 2 seconds of being commanded.

REQUIREMENTS

Response Requirement

SYSTEM UNDER TEST

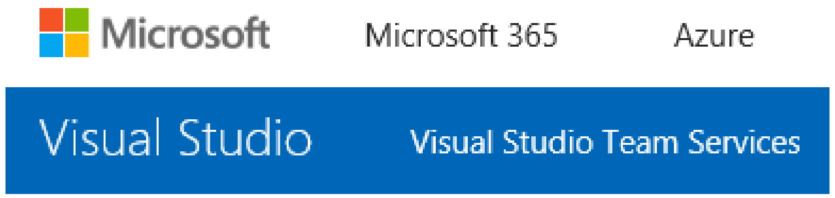
Model: FlutterSuppressionSystem_IO_Harness

Automated Testing with Simulink Test

Real-Time Test Automation, ideal for Hardware-in-the-Loop

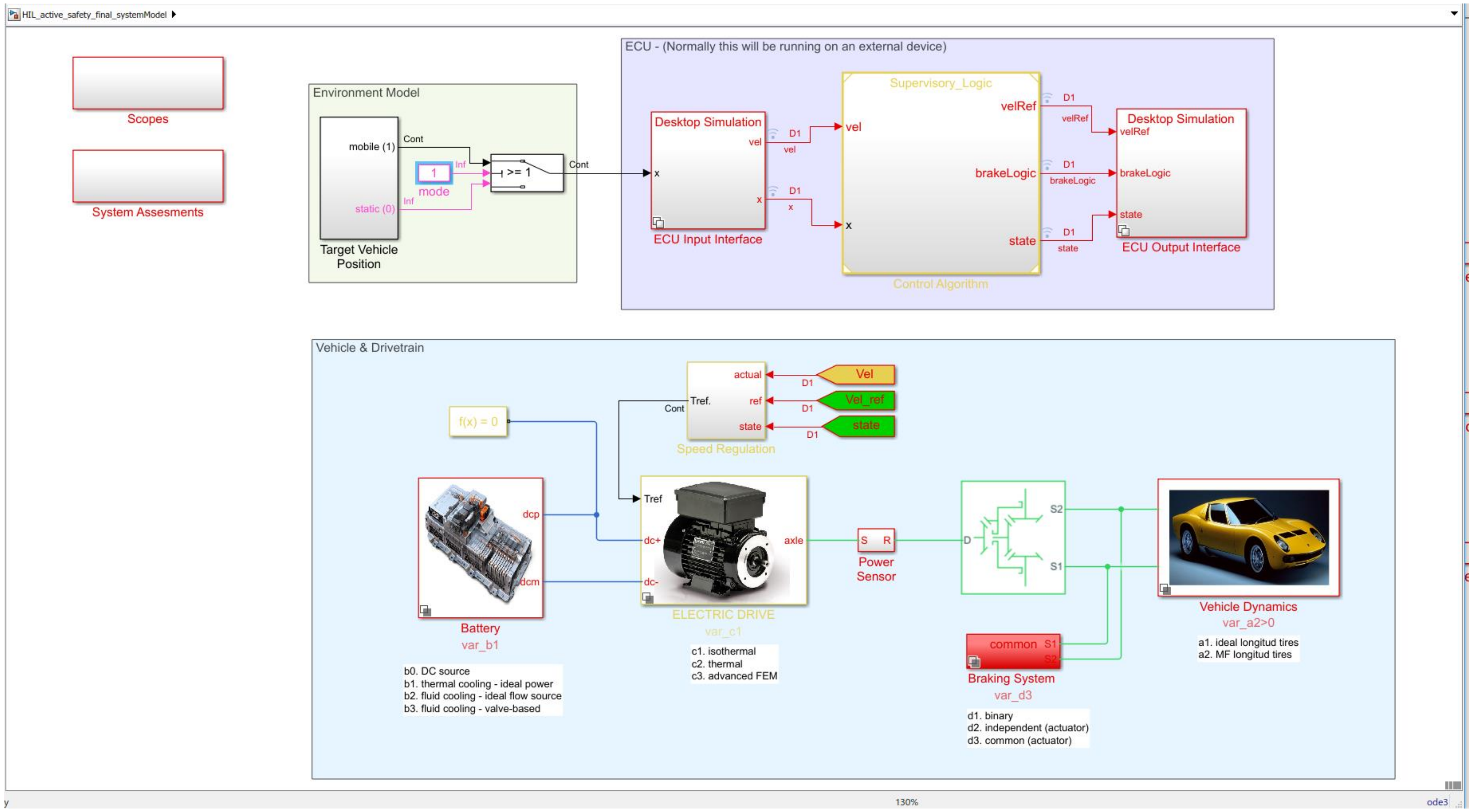
- Integrates with MATLAB Unit Test
- Supports the TAP protocol to run from CI systems like Jenkins, VSTS, etc.

,



, ...

Demo!



HIL_active_safety_final_systemModel
Scopes
System Assesments
Environment Model
mobile (1)
static (0)
Cont
Inf
1
mode
>= 1
Target Vehicle Position
ECU - (Normally this will be running on an external device)
Desktop Simulation
vel
x
ECU Input Interface
D1
Supervisory_Logic
velRef
brakeLogic
state
Control Algorithm
Desktop Simulation
ECU Output Interface
Vehicle & Drivetrain
actual
Tref.
ref
state
Speed Regulation
Vel
Vel_ref
state
f(x) = 0
dcp
dcm
Battery
var_b1
b0. DC source
b1. thermal cooling - ideal power
b2. fluid cooling - ideal flow source
b3. fluid cooling - valve-based
Tref
dc+
dc-
axle
ELECTRIC DRIVE
var_c1
c1. isothermal
c2. thermal
c3. advanced FEM
S R
Power Sensor
D
S2
S1
common
Braking System
var_d3
d1. binary
d2. independent (actuator)
d3. common (actuator)
Vehicle Dynamics
var_a2>0
a1. ideal longitud tires
a2. MF longitud tires
130%
ode3

Key Take-Aways From This Presentation

- Agile = iterative and short design cycles, high degree of re-use of digital assets is essential

- Technology landscape demands faster control loops, prototyping with SW is not enough anymore

Questions?

Thank You!