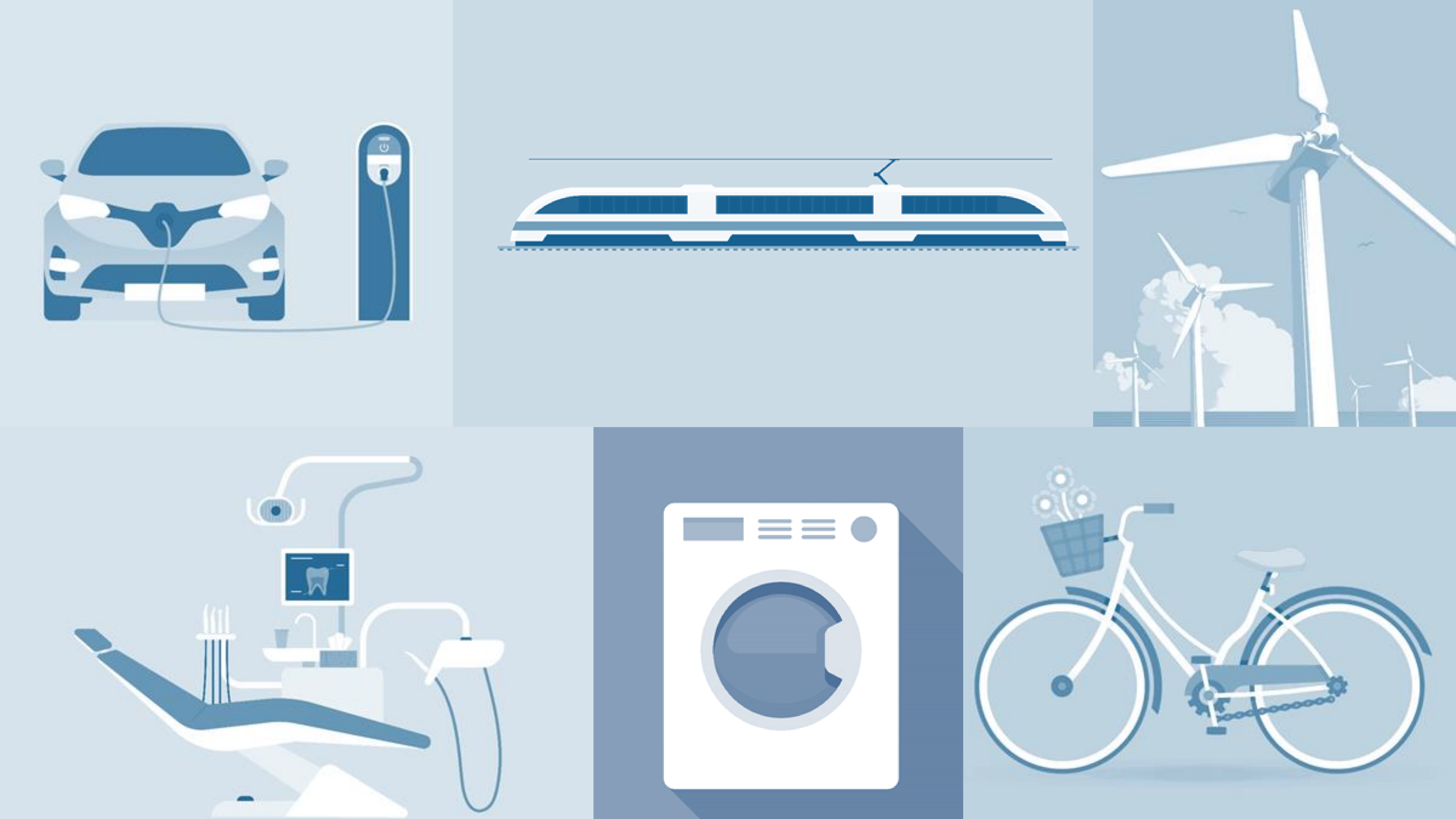
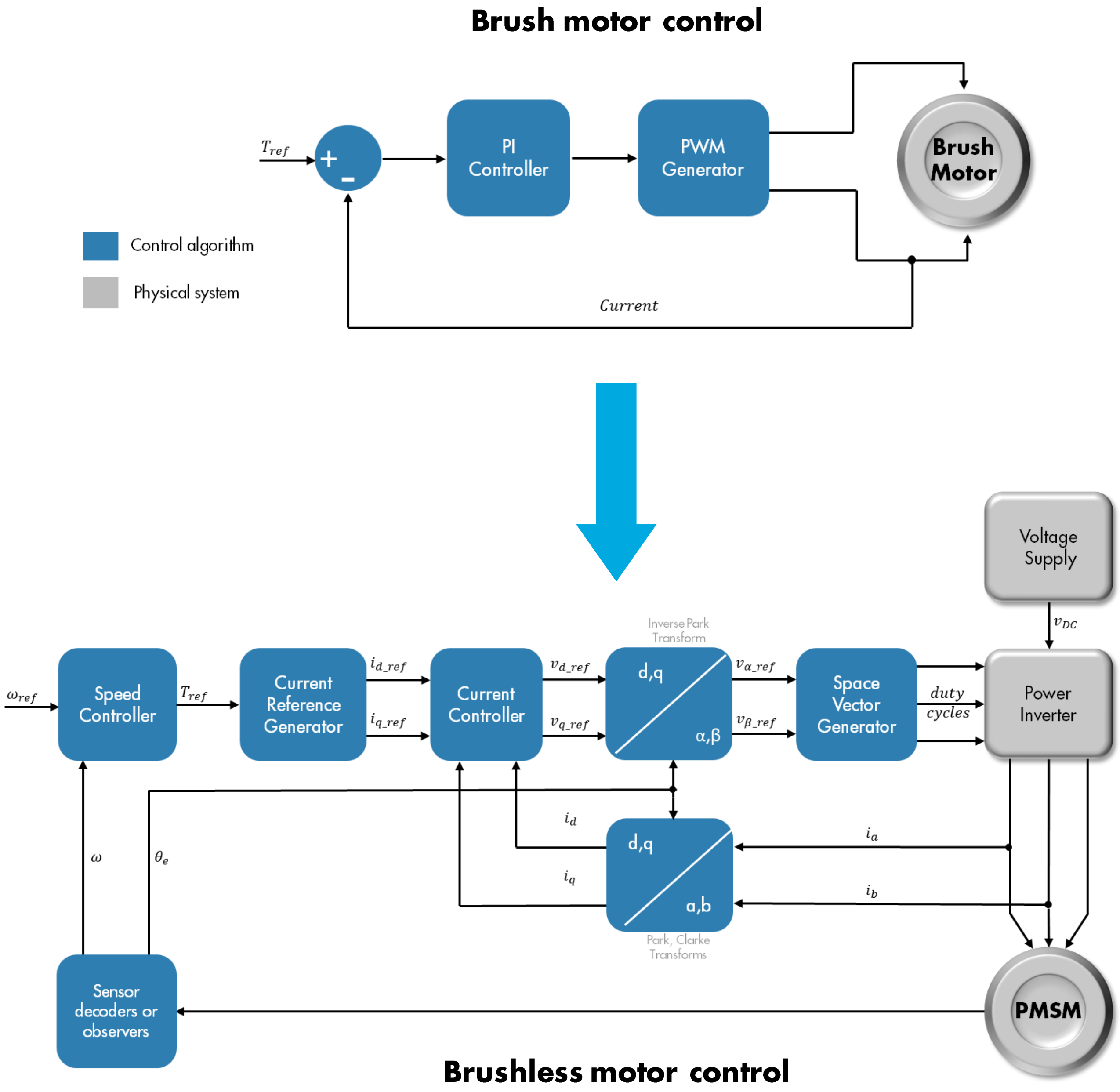
Développement d’un FOC pour PMSM

Accélérer l’électrification



More Complex Algorithms

- Increasing motor control algorithm **complexity**
  - Field-Oriented Control (FOC)
  - Field-weakening control
  - Sensorless
  - Space vector PWM
- Increasing need to run these algorithms **faster**
  - Wide bandgap semiconductors
  - Increasing popularity of motor types such as switched reluctance motors



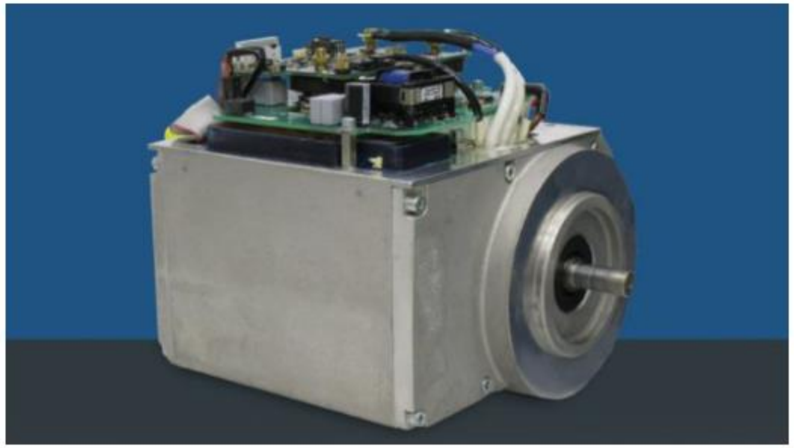
Model-Based Design

Answer to complex design

ATB Technologies Cuts Electric Motor Controller Development Time by 50% Using Code Generation for TI's C2000 MCU

"MathWorks tools enabled us to verify the quality of our design at multiple stages of development, and to produce a high-quality component within a short time frame."

— *Markus Schertler, ATB Technologies*



ATB Technologies permanent magnet synchronous motor.

Challenge

Develop control software to maximize the efficiency and performance of a permanent magnet synchronous motor

Solution

Use MathWorks tools for Model-Based Design to model, simulate, and implement the control system on a target processor

Results

- Development time cut in half
- Design reviews simplified
- Target verification and deployment accelerated

ITK Engineering Develops IEC 62304–Compliant Controller for Dental Drill Motor with Model-Based Design

"Model-Based Design with Simulink enabled us to reduce costs and project risk through early verification, shorten time to market on an IEC 62304–certified system, and deliver high-quality production code that was first-time right."

— *Michael Schwarz, ITK Engineering*

Dental drills featuring ITK Engineering's sensorless brushless motor control.

Challenge

Develop and implement field-oriented controller software for sensorless brushless DC motors for use in dental drills

Solution

Use Model-Based Design with Simulink, Stateflow, and Embedded Coder to model the controller and plant, run closed-loop simulations, generate production code, and streamline unit testing

Results

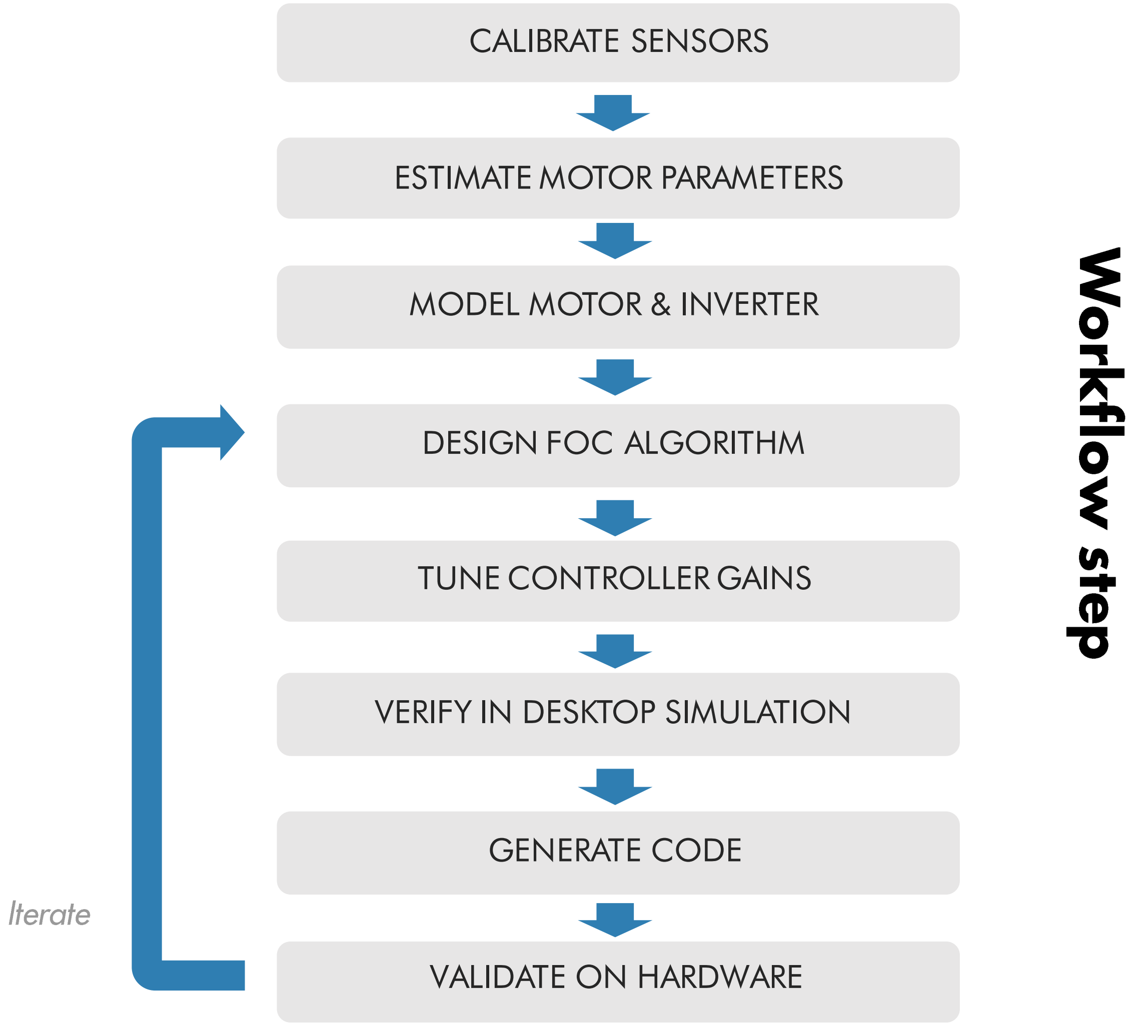
- Development time halved
- Hardware problems discovered early
- Contract won, client confidence established

Why Simulink for motor control?

Motor Control Blockset

- Verify control algorithm with desktop simulation
- Generate compact and fast code from models
- Minimize development time using reference examples

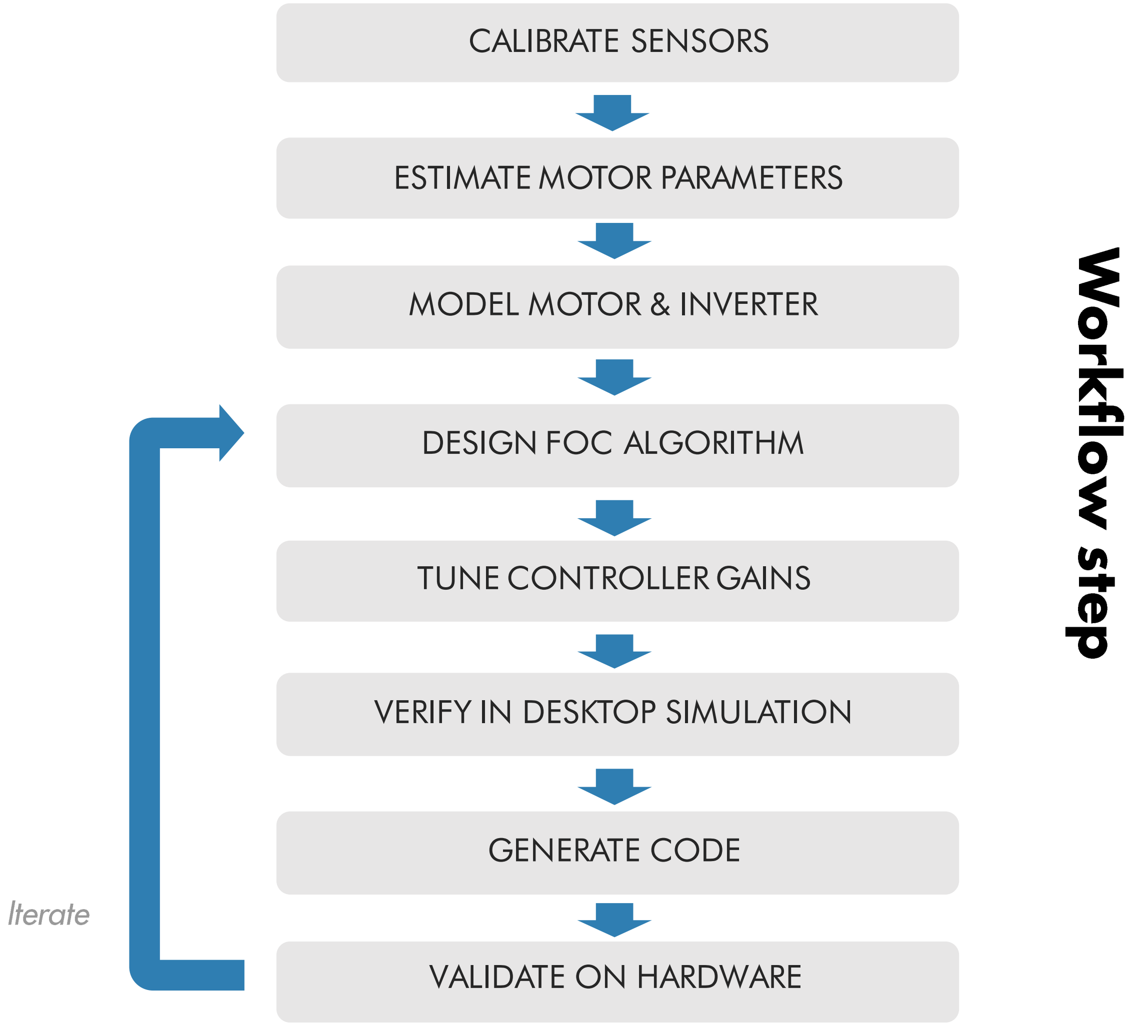
➔ Customers routinely report
50% faster time to market


CALIBRATE SENSORS
ESTIMATE MOTOR PARAMETERS
MODEL MOTOR & INVERTER
DESIGN FOC ALGORITHM
TUNE CONTROLLER GAINS
VERIFY IN DESKTOP SIMULATION
GENERATE CODE
VALIDATE ON HARDWARE
Iterate
Workflow step

Agenda

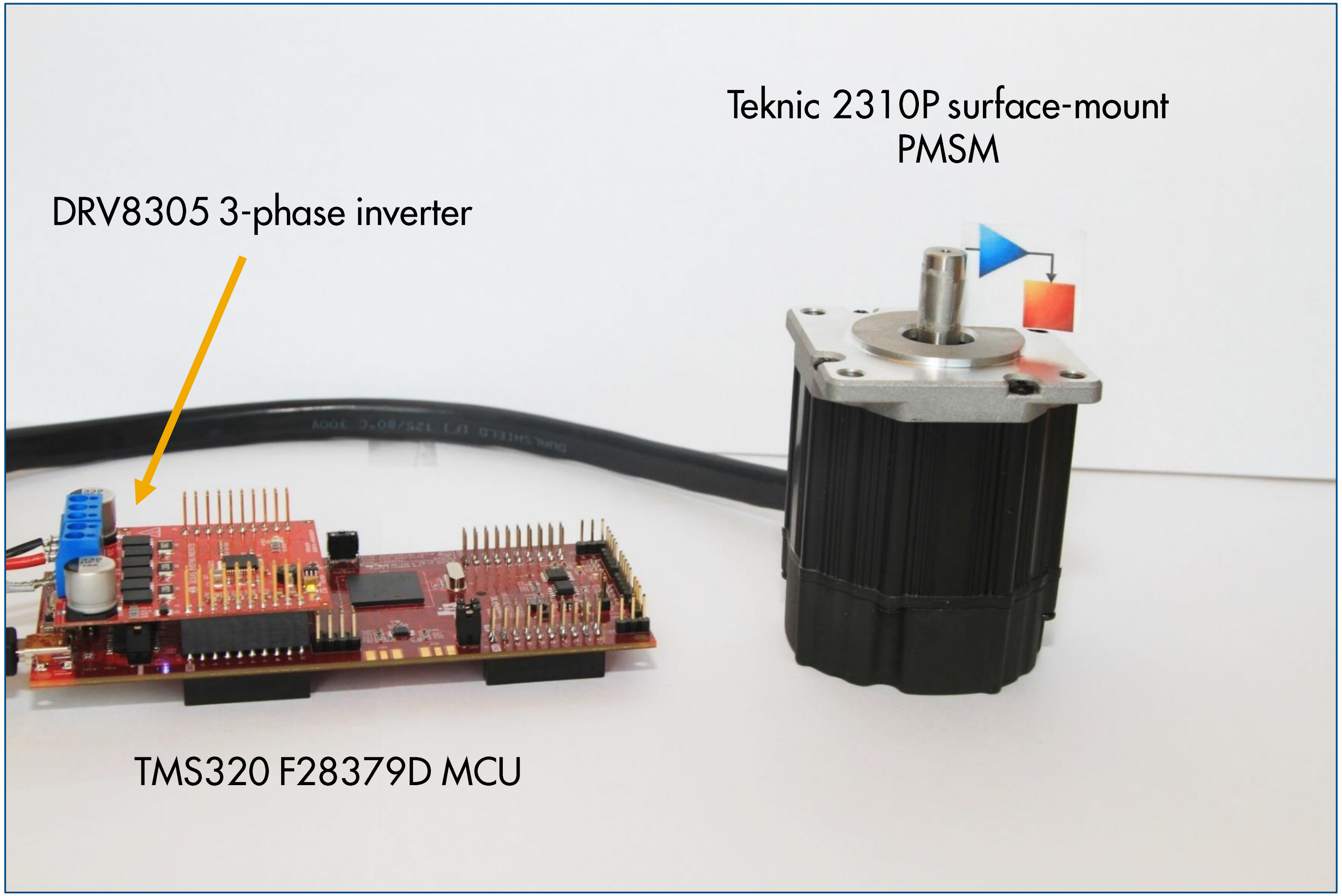
From Desktop Simulation to Software Deployment

- Plant modeling
  - Sensors Calibration
  - Motor Parameters Estimation
  - Motor and Inverter Model
- Algorithm design with simulation
  - Field-Oriented control
  - Autotuning control gain
  - Verifying controller
- Software deployment
  - Code generation



Texas Instruments Motor Control Kit

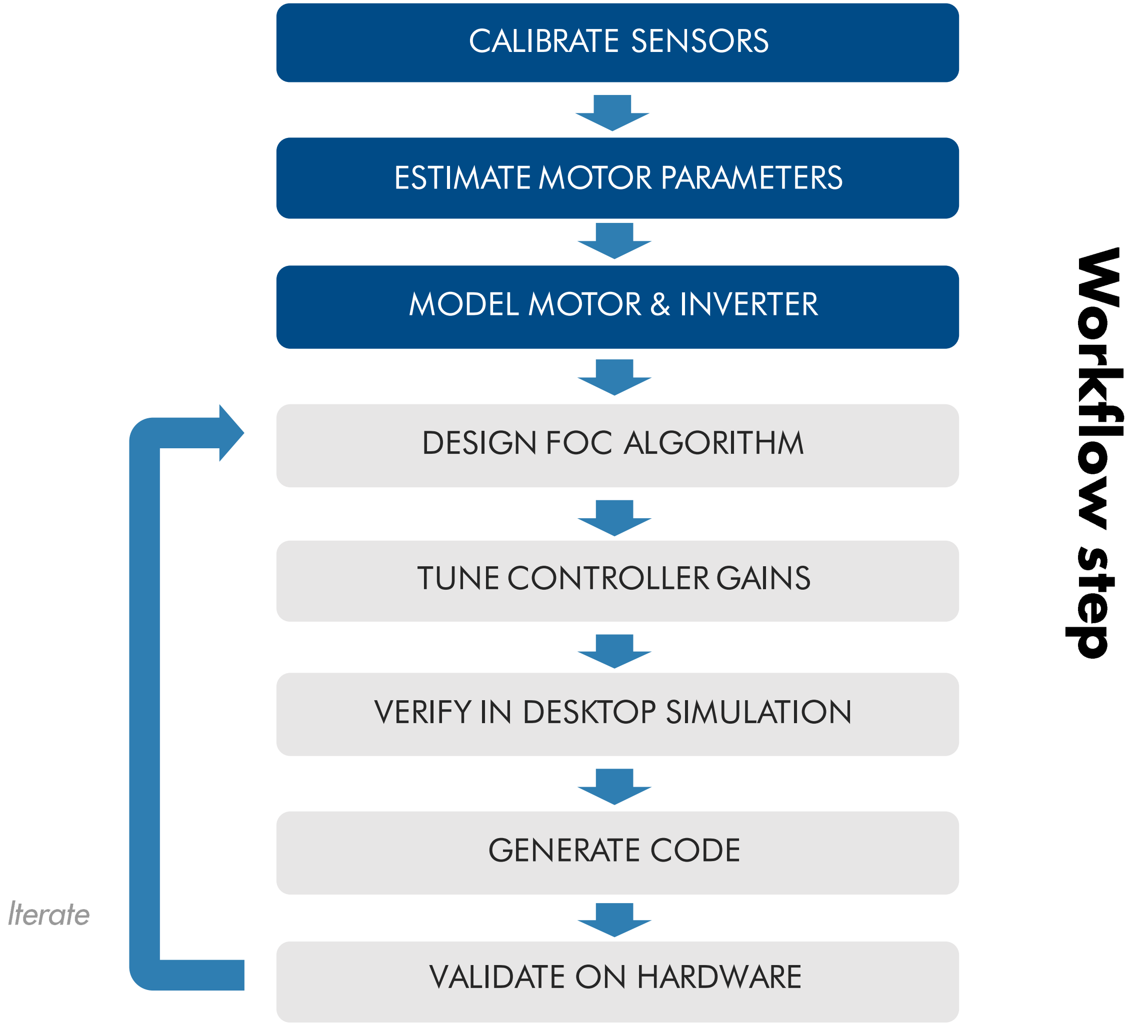
Hardware configuration



Agenda

From Desktop Simulation to Software Deployment

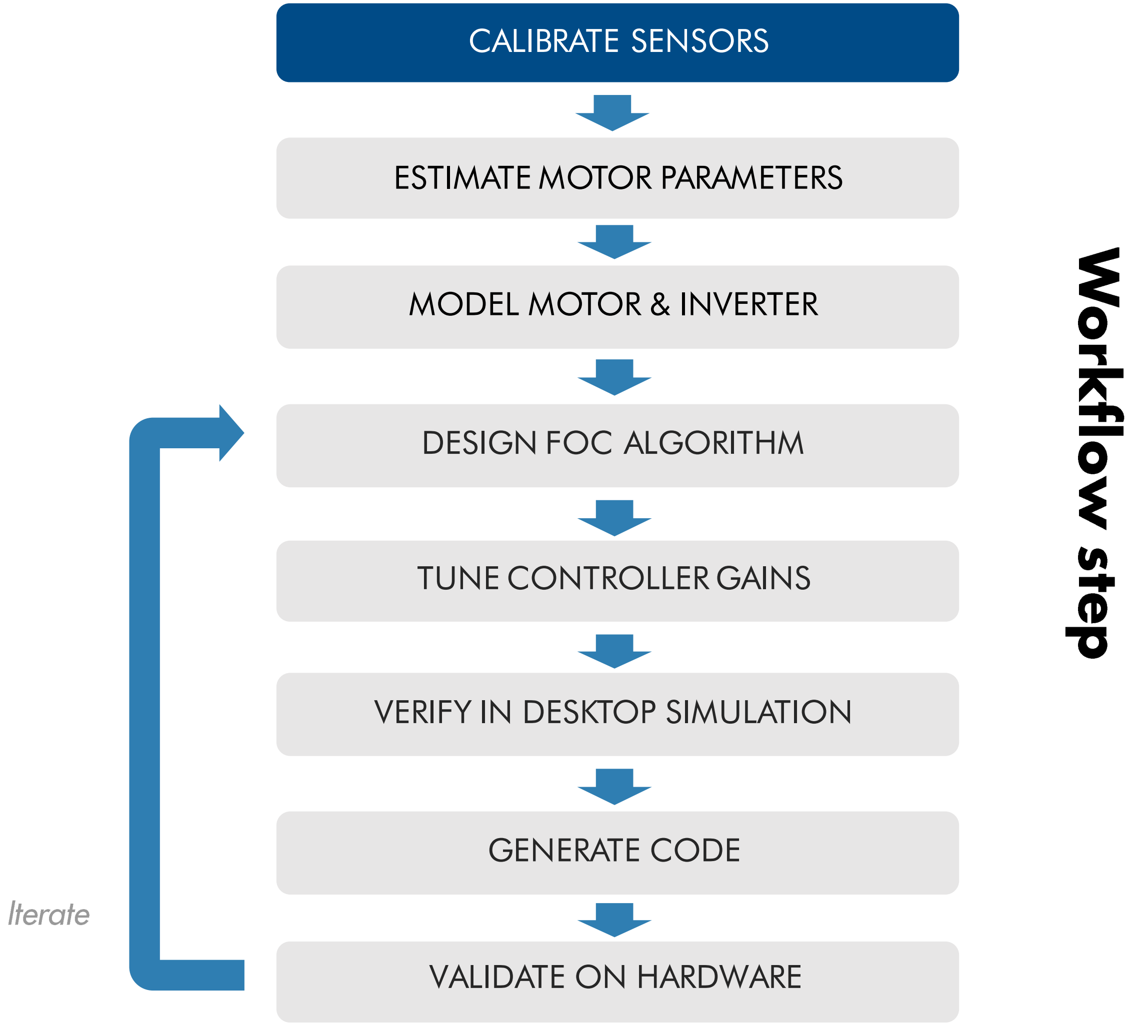
- Plant modeling
  - Sensors Calibration
  - Motor Parameters Estimation
  - Motor and Inverter Model
- Algorithm design with simulation
  - Field-Oriented control
  - Autotuning control gain
  - Verifying controller
- Software deployment
  - Code generation



Agenda

From Desktop Simulation to Software Deployment

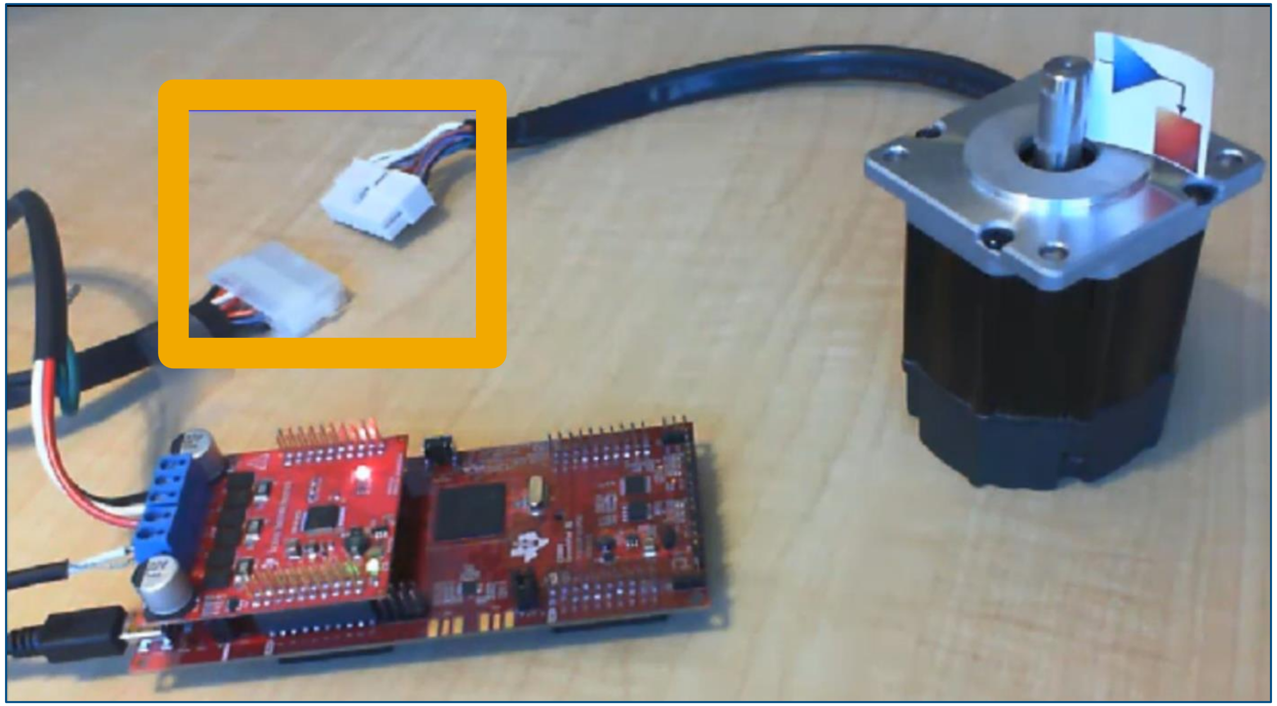
- Plant modeling
  - Sensors Calibration
  - Motor Parameters Estimation
  - Motor and Inverter Model
- Algorithm design with simulation
  - Field-Oriented control
  - Autotuning control gain
  - Verifying controller
- Software deployment
  - Rapid control prototyping
  - Code generation
  - Hardware-In-The-Loop (HIL) test



Sensor Calibration

Plant Modeling

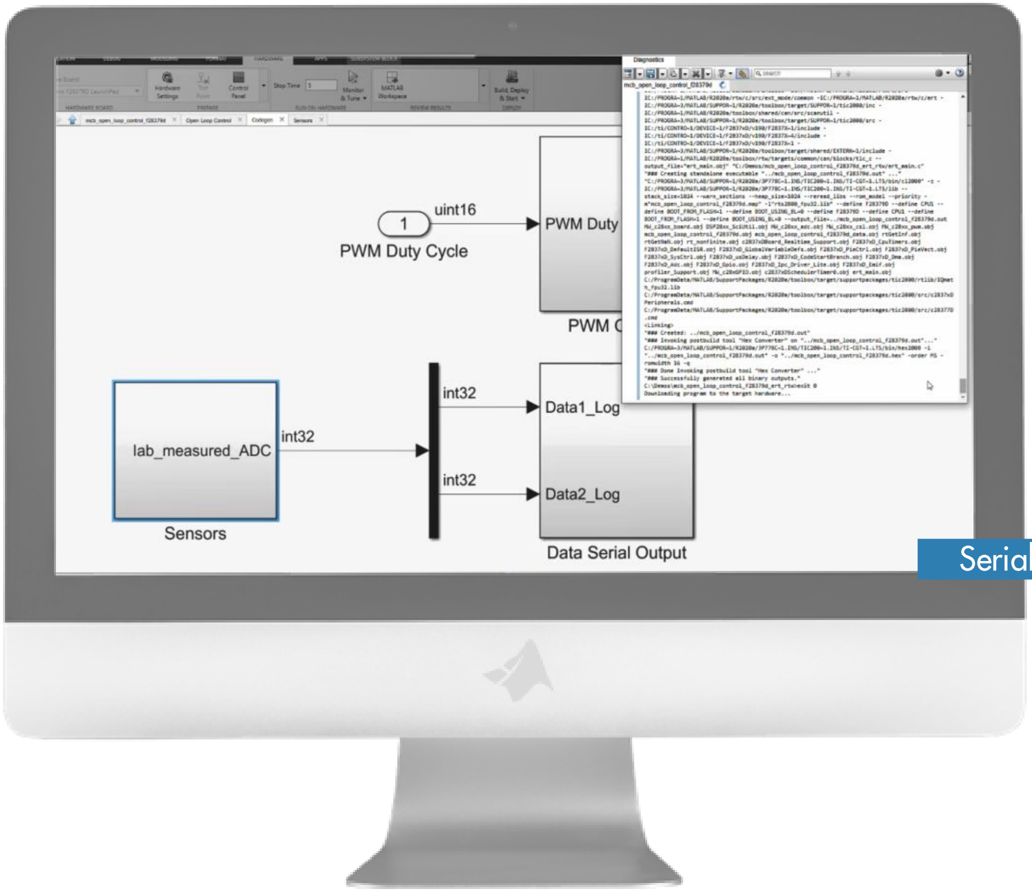
- ADC offsets
- Position Sensor Offset



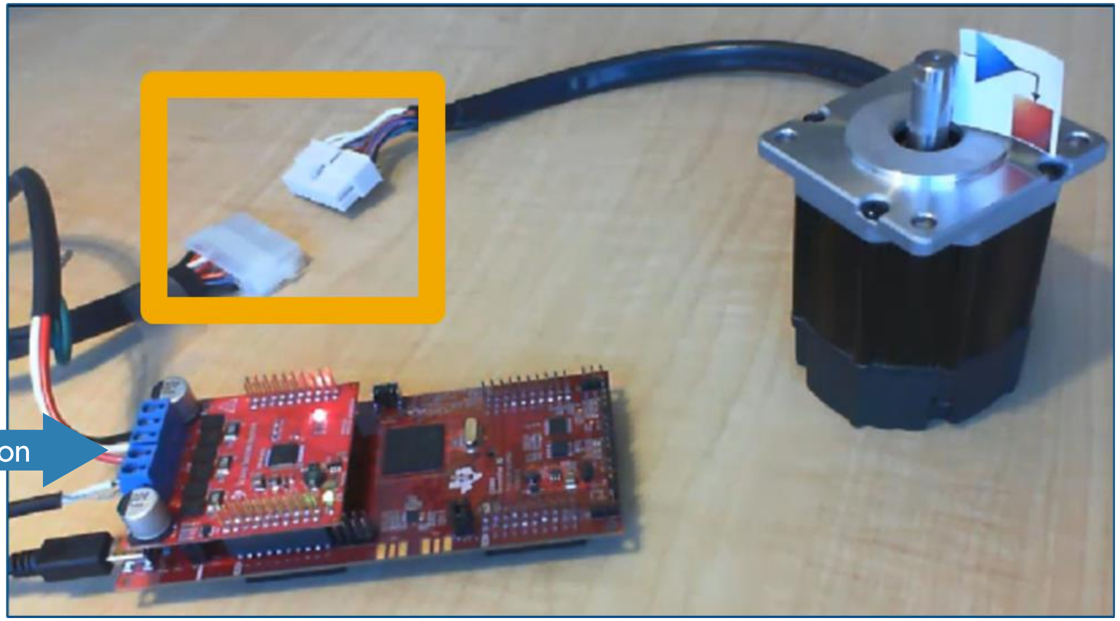
ADC Offsets

Plant Modeling

3 Code Generation / Compile and Link / Ready to Run



Serial connection



Development Computer with ADC offset calibration model

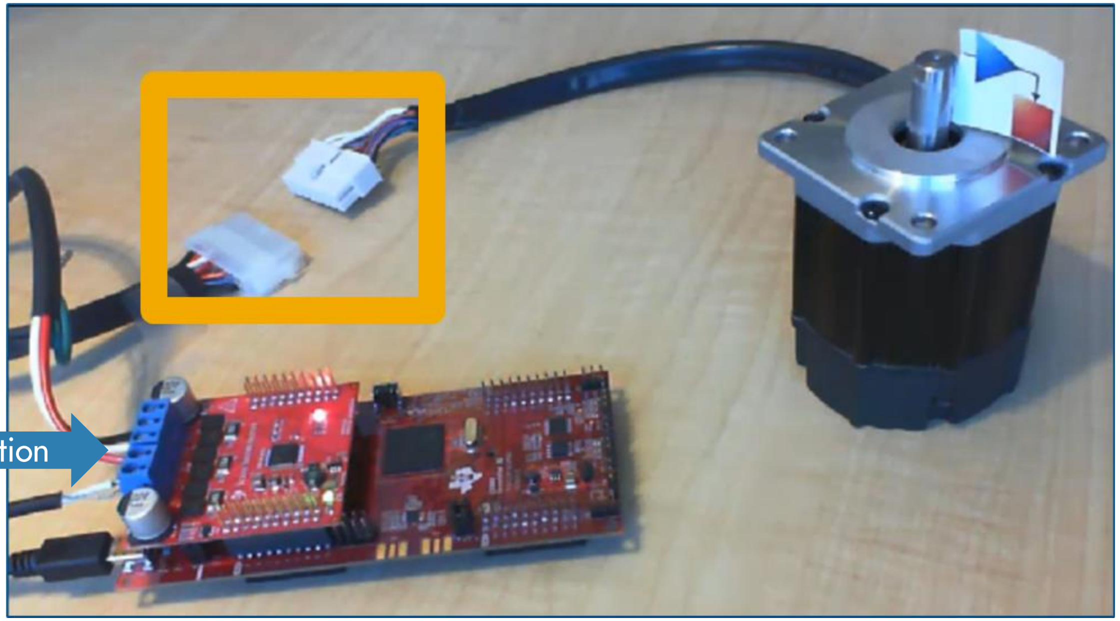
Target Hardware

ADC Offsets

Plant Modeling

4 External Mode Simulation

Serial connection



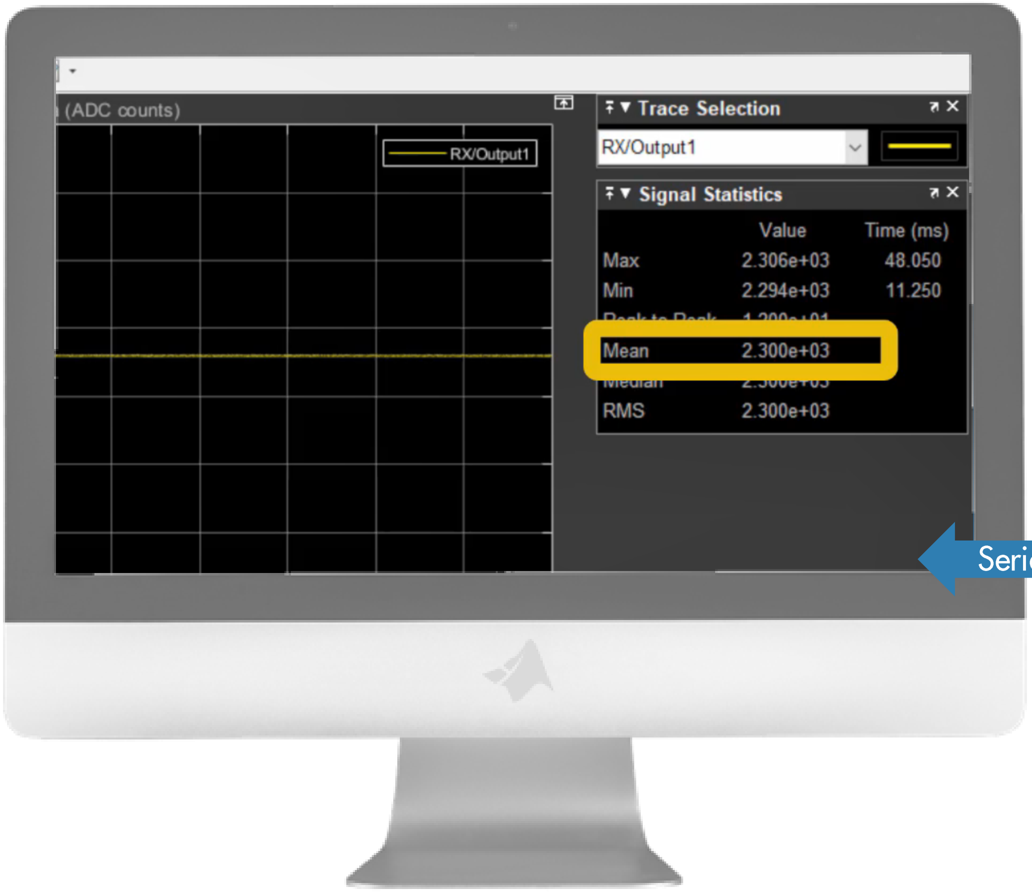
Development Computer with ADC offset calibration model

Target Hardware

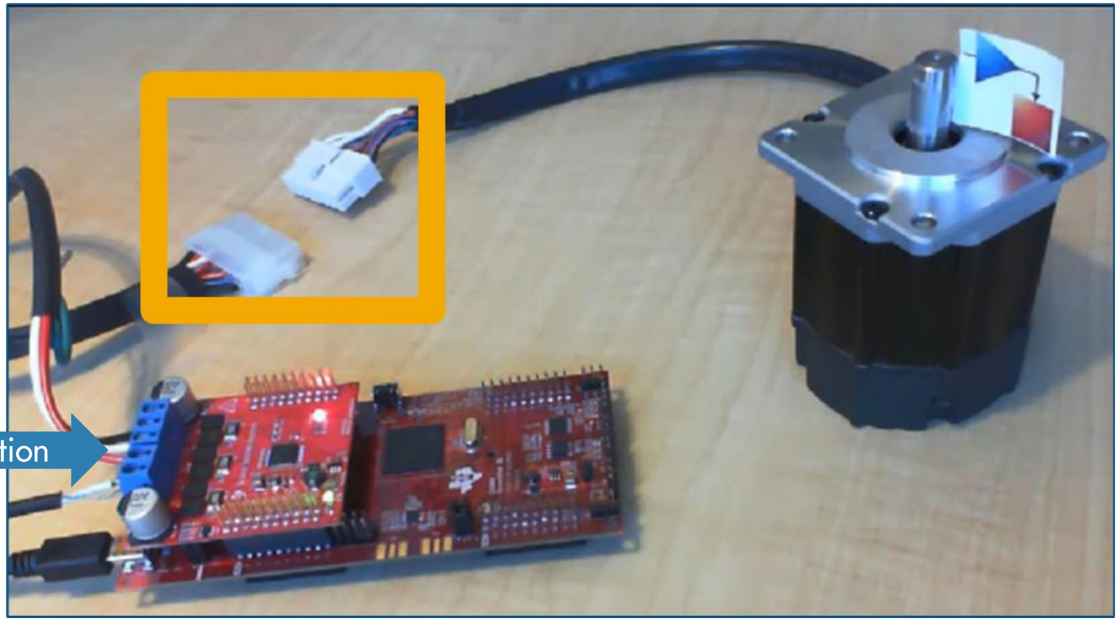
ADC Offsets

Plant Modeling

5 Get Offset for Phase A and B



Serial connection



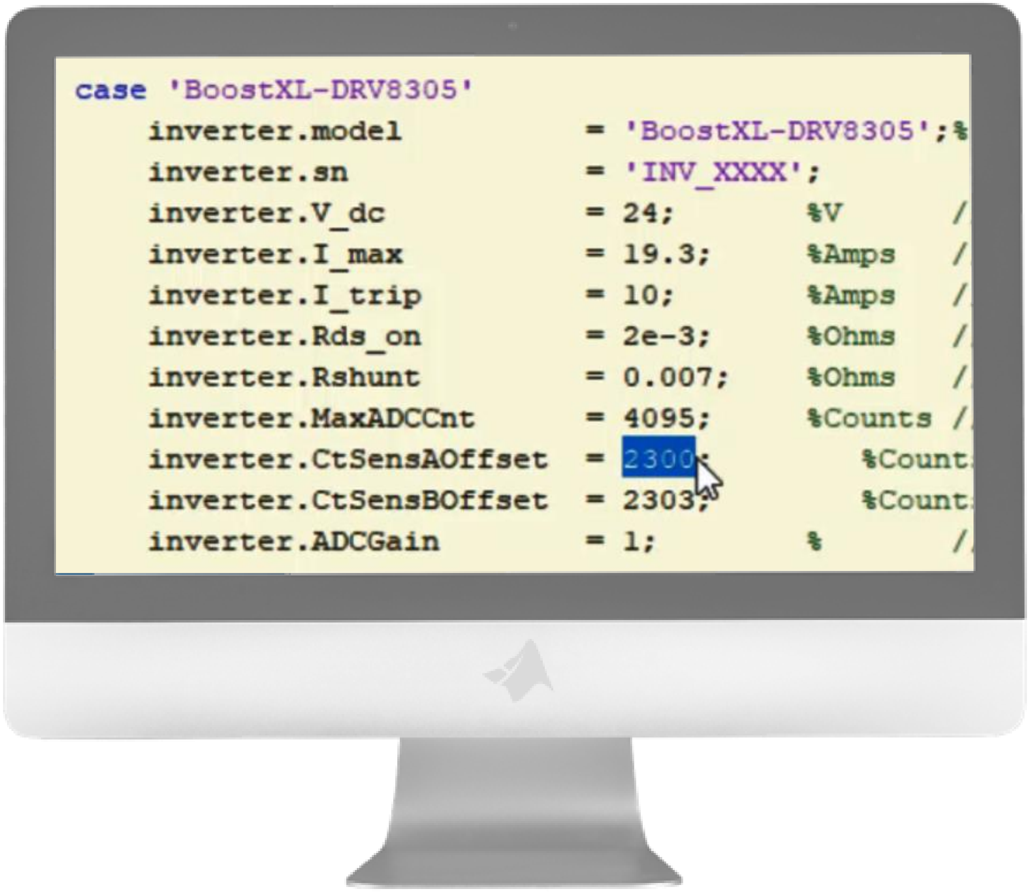
Development Computer with ADC offset calibration model

Target Hardware

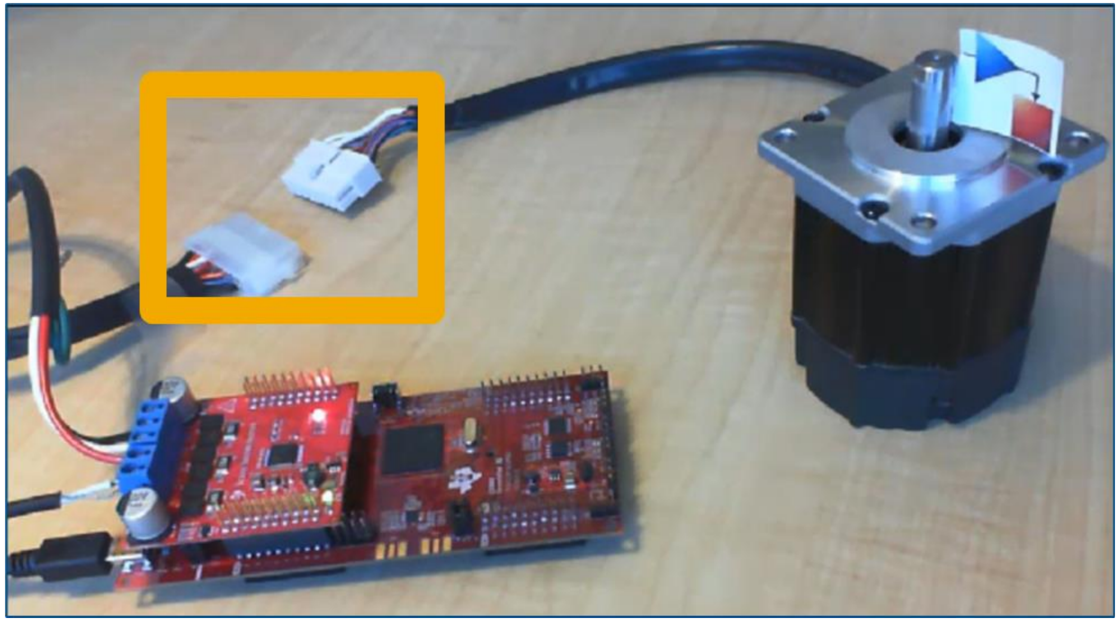
ADC Offsets

Plant Modeling

6 Parameterization script



Development Computer with ADC offset calibration model



Target Hardware

Position Sensor Offset

Plant Modeling

Documentation

Search R2021a Documentation

CONTENTS

All | Examples | Blocks

• Documentation Home

Sensor Calibration

Hall Offset Calibration for PMSM Motor

Calculates the offset between the rotor direct axis (d-axis) and position detected by the Hall sensor. The field-oriented control (FOC)

Open Example

Quadrature Encoder Offset Calibration for PMSM Motor

Calculates the offset between the d-axis of the rotor and encoder index pulse position as detected by the quadrature encoder sensor. The

Open Example

Hall Offset Calibration for PMSM Motor

R2021a

Open Example

This example calculates the offset between the rotor direct axis (d-axis) and position detected by the Hall sensor. The field-oriented control (FOC) algorithm needs this position offset to run the permanent magnet synchronous motor (PMSM) correctly. To compute the offset, the target model runs the motor in the open-loop condition. The model uses a constant V_d (voltage along the stator's d-axis) and a zero V_q (voltage along the stator's q-axis) to run the motor (at a low constant speed) by using a position or ramp generator. When the position or ramp value reaches zero, the corresponding rotor position is the offset value for the Hall sensors.

The control algorithm (available in the field-oriented control and parameter estimation examples) uses this offset value to compute an accurate position of d-axis of the rotor. The controller needs this offset to optimally run the PMSM.

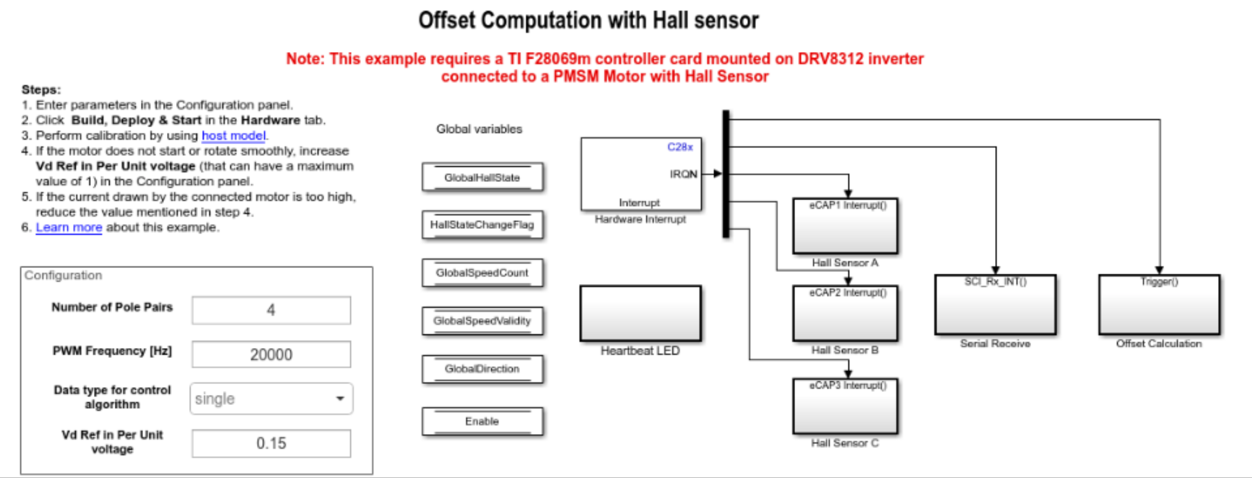
Models

This example includes these models:

- mcb_pmsm_hall_offset_f28069m
- mcb_pmsm_hall_offset_f28379d

You can use these models only for code generation. You can also use the open_system command to open the Simulink® models. For example, use this command for a F28069M based controller:

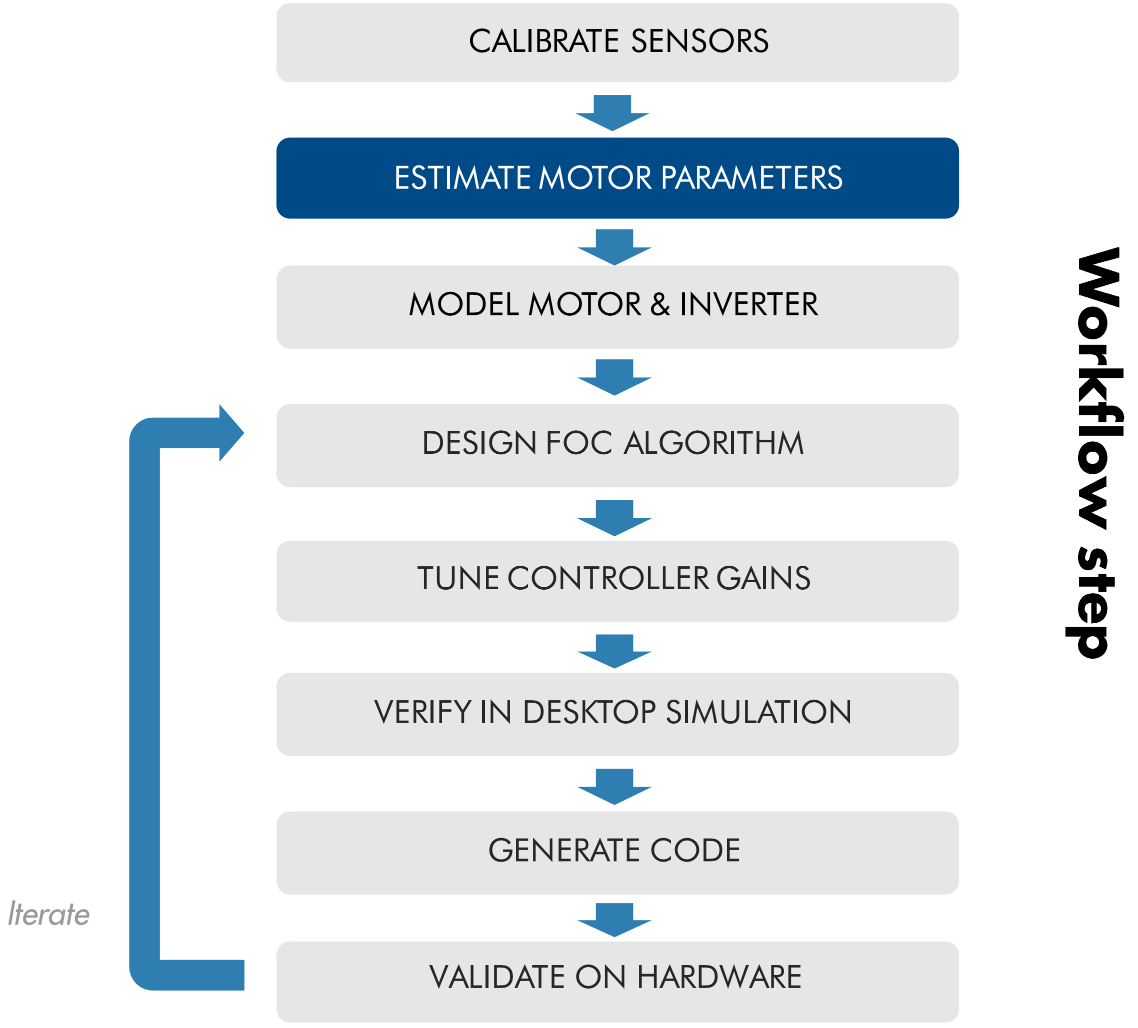
```
open_system('mcb_pmsm_hall_offset_f28069m.slx');
```



Agenda

From Desktop Simulation to Software Deployment

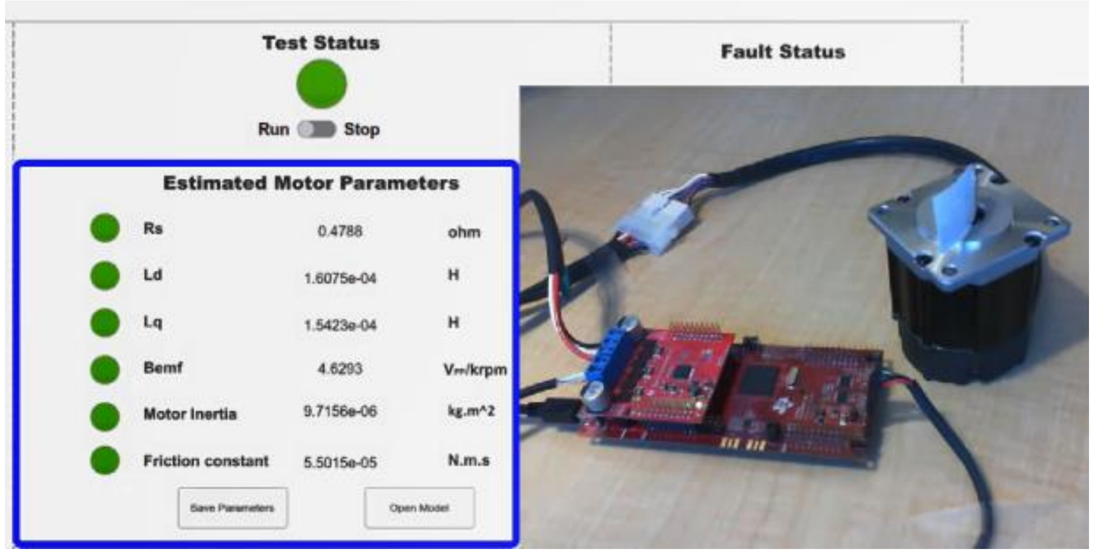
- Plant modeling
  - Sensors Calibration
  - Motor Parameters Estimation
  - Motor and Inverter Model
- Algorithm design with simulation
  - Field-Oriented control
  - Autotuning control gain
  - Verifying controller
- Software deployment
  - Rapid control prototyping
  - Code generation
  - Hardware-In-The-Loop (HIL) test



Motor Parameters Estimation

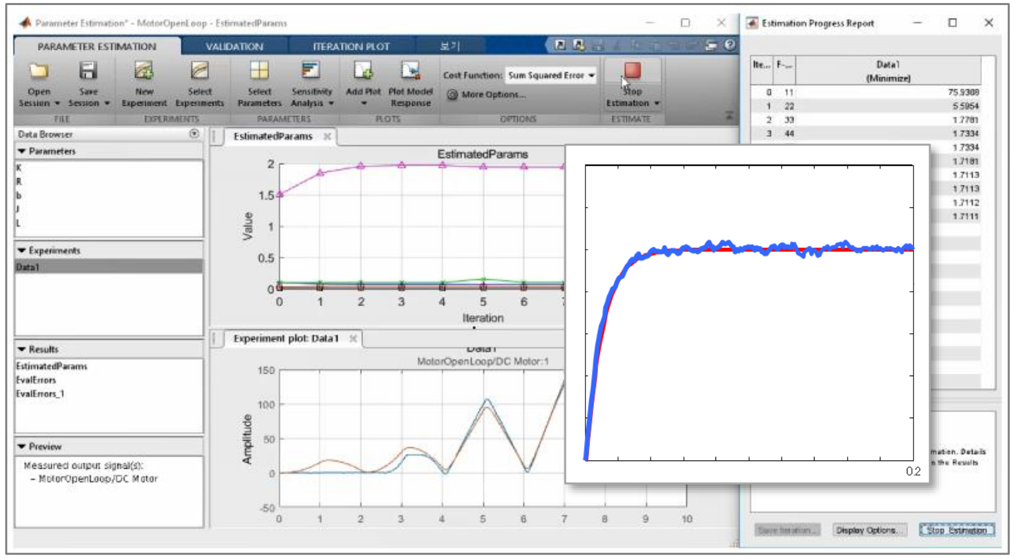
Plant Modeling

Two types of parameter estimation methods:



Parameter Estimation with Instrumented Test

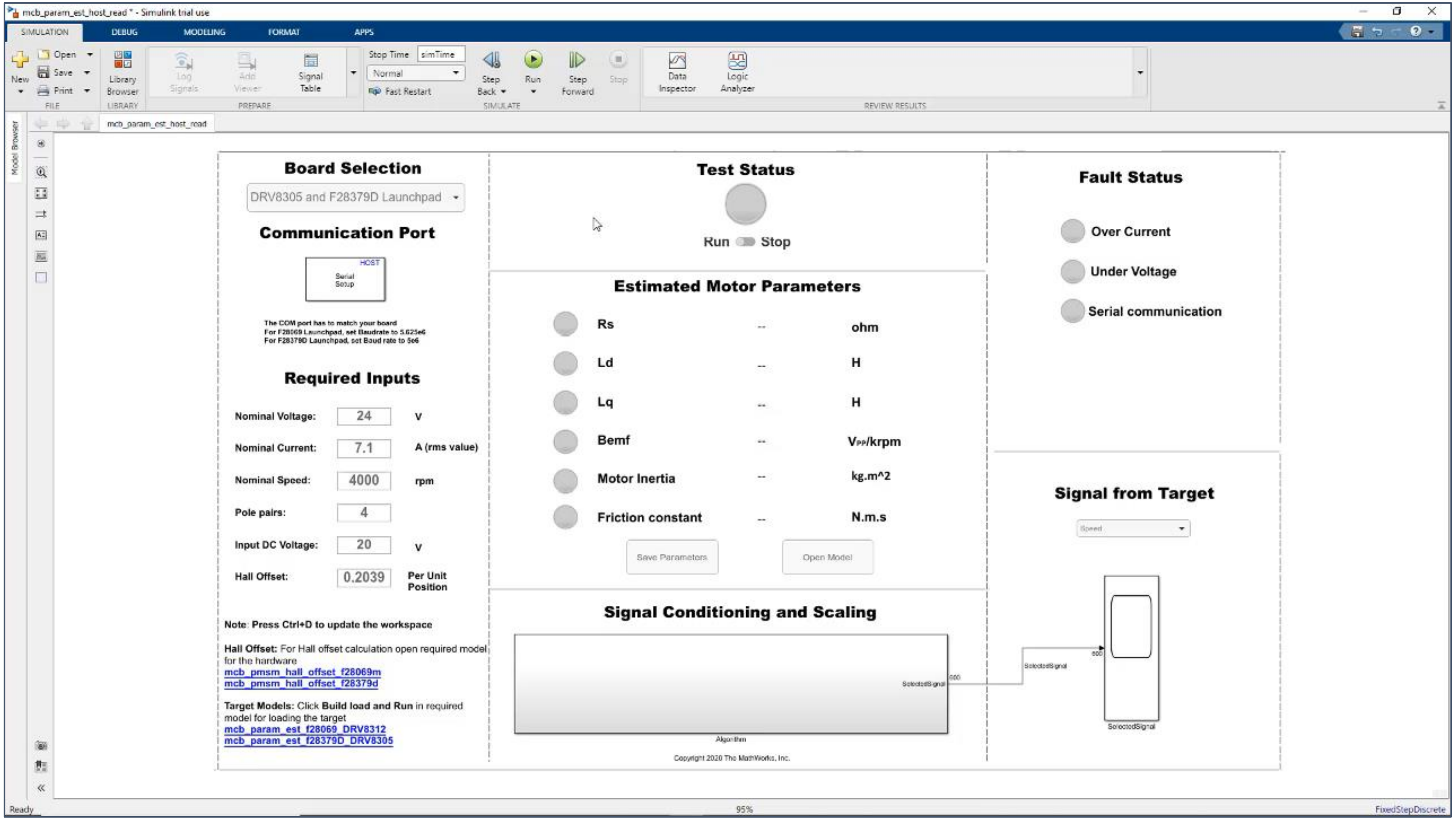
or



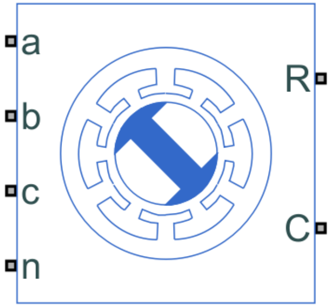
Parameter Estimation using Operation Data

Motor Parameters Estimation - Instrumented Test

Plant Modeling

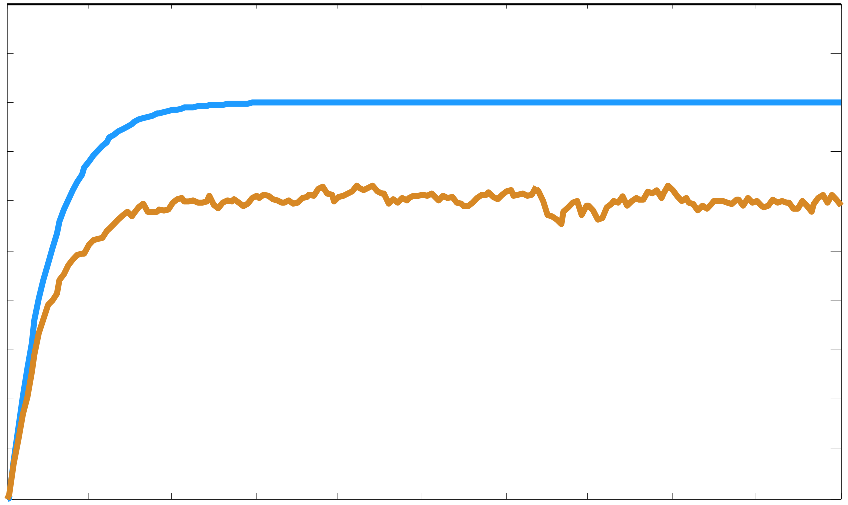


Parameter Estimation Using Operation Data



PMSM model

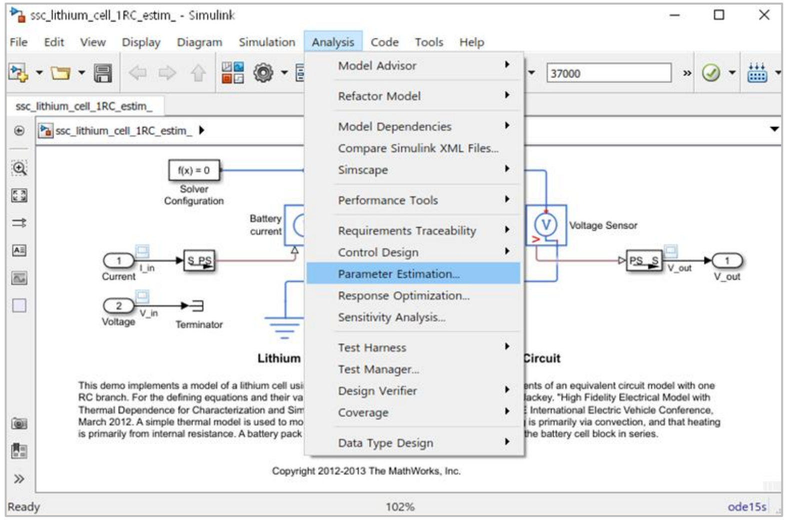
Real PMSM



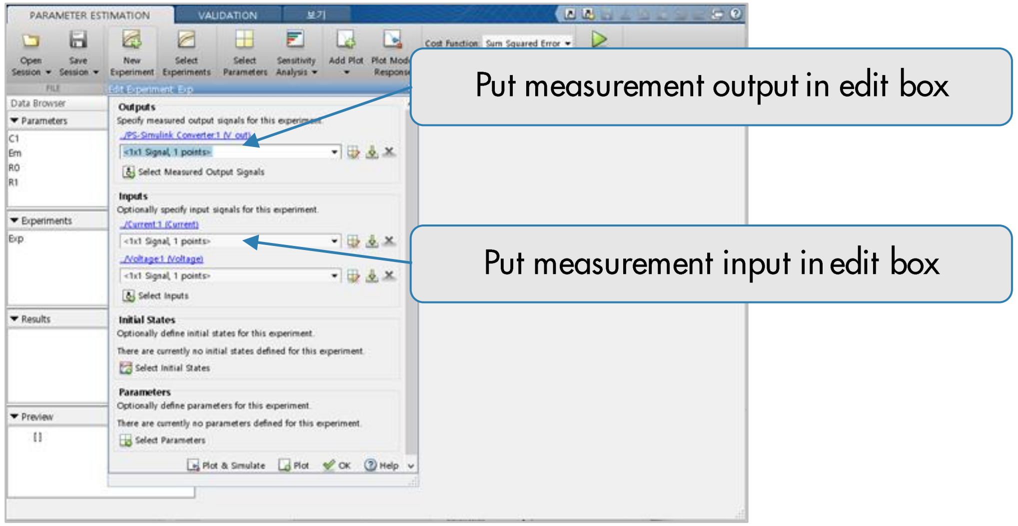
Problem: Simulation data does not match measured data because the parameters are incorrect

Solution: Use **Simulink Design Optimization** to automatically tune model parameters

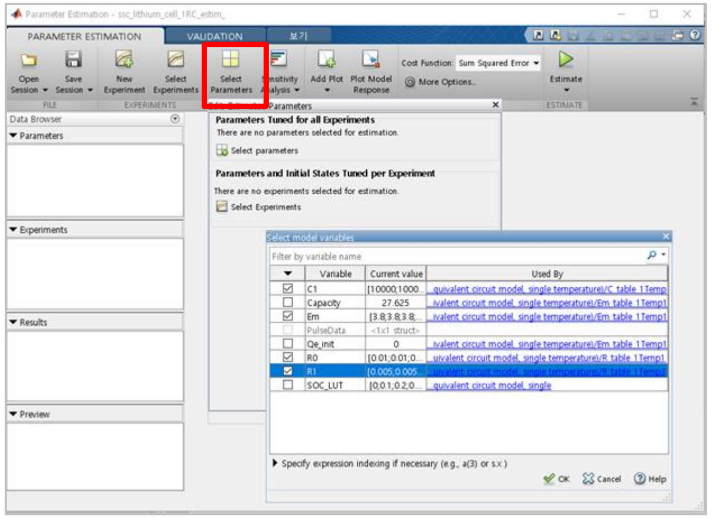
1 Open Parameter Estimation



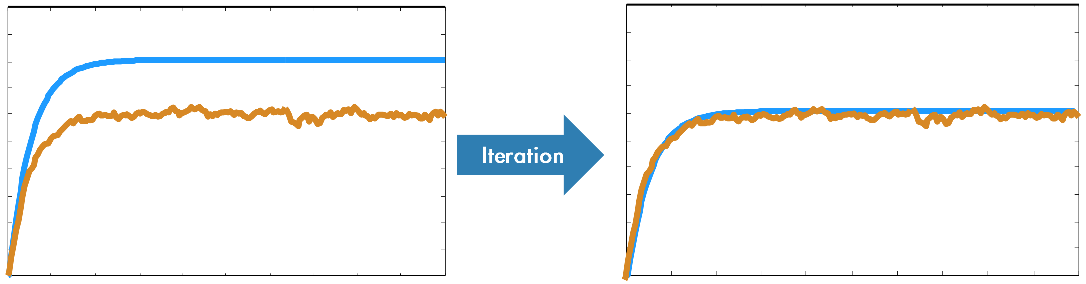
2 Make New Experiment



3 Select Tuning Parameters



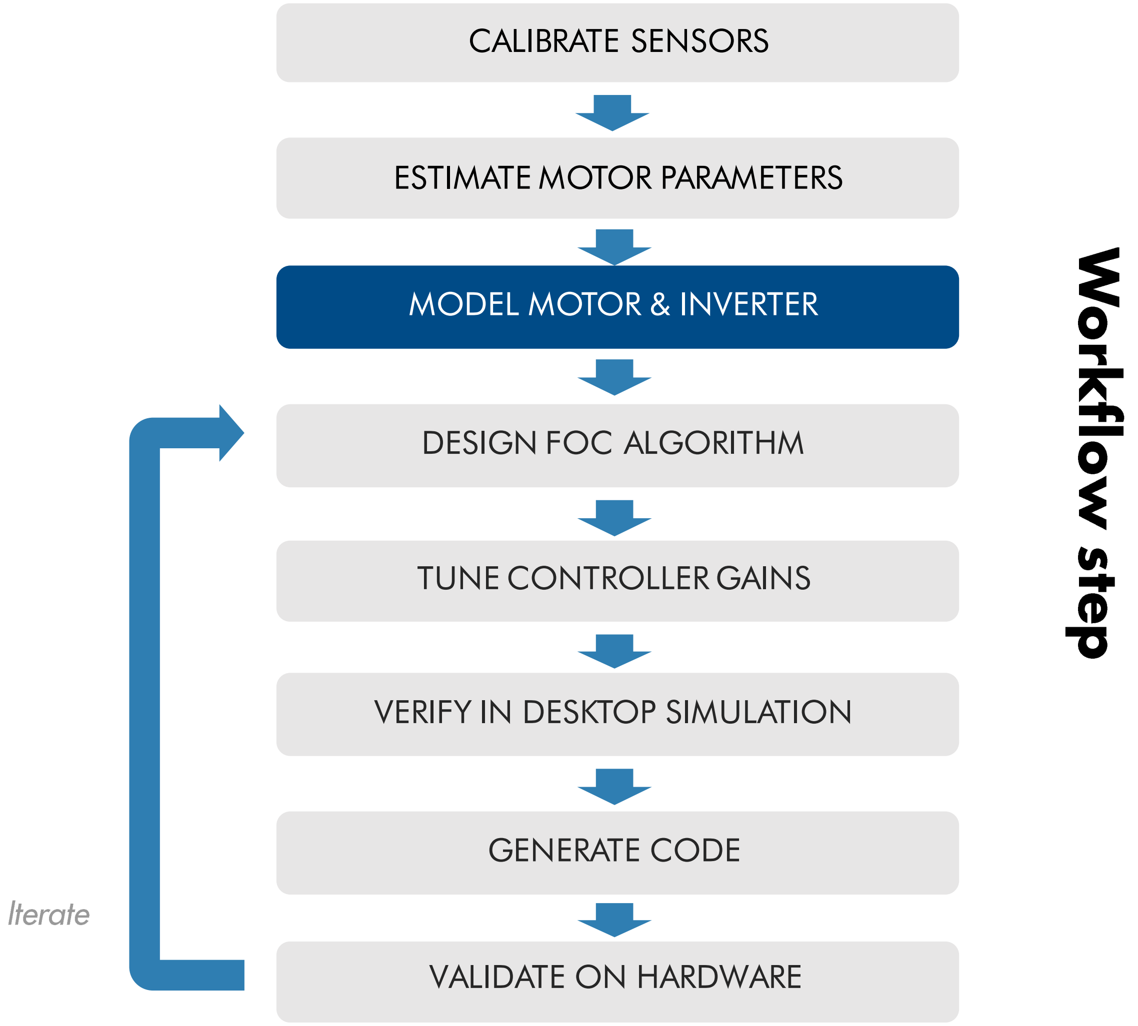
4 Run Optimization



Agenda

From Desktop Simulation to Software Deployment

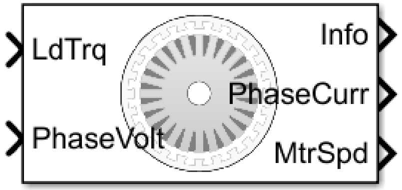
- Plant modeling
  - Sensors Calibration
  - Motor Parameters Estimation
  - Motor and Inverter Model
- Algorithm design with simulation
  - Field-Oriented control
  - Autotuning control gain
  - Verifying controller
- Software deployment
  - Rapid control prototyping
  - Code generation
  - Hardware-In-The-Loop (HIL) test



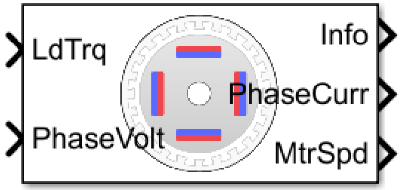
Motor and Inverter Modeling

Choose the right level of fidelity

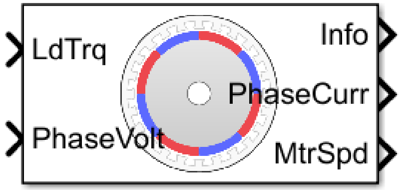
- Use linear lumped-parameter model shipped with **Motor Control Blockset**



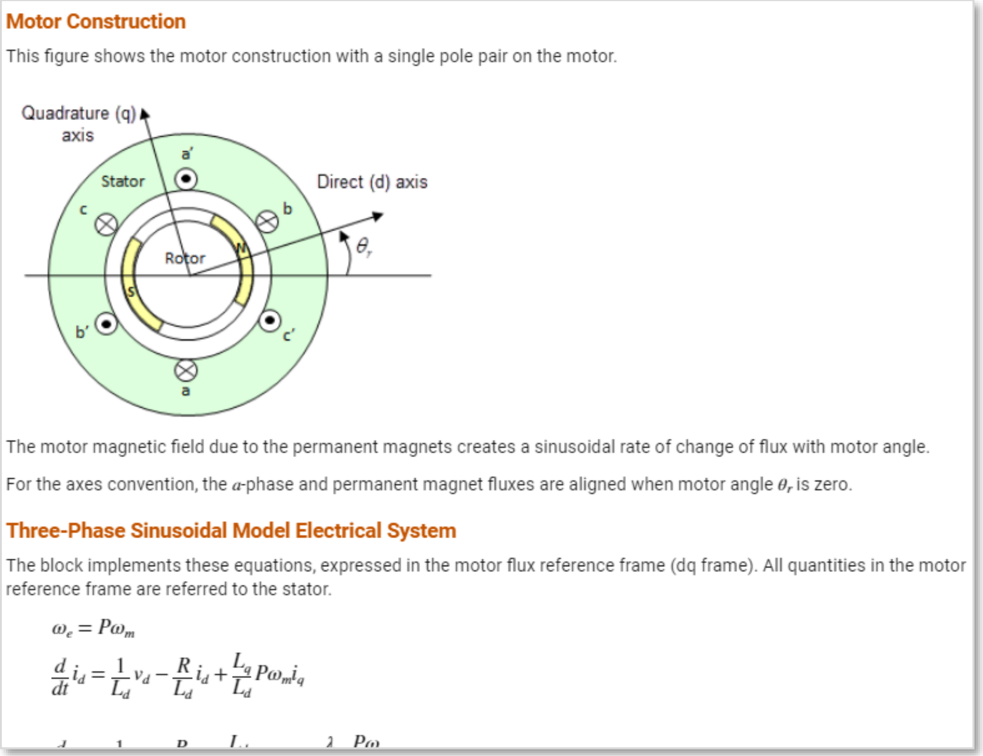
Induction Motor



Interior PMSM



Surface Mount PMSM


Motor Construction

This figure shows the motor construction with a single pole pair on the motor.

The motor magnetic field due to the permanent magnets creates a sinusoidal rate of change of flux with motor angle.

For the axes convention, the *a*-phase and permanent magnet fluxes are aligned when motor angle θ_r is zero.

Three-Phase Sinusoidal Model Electrical System

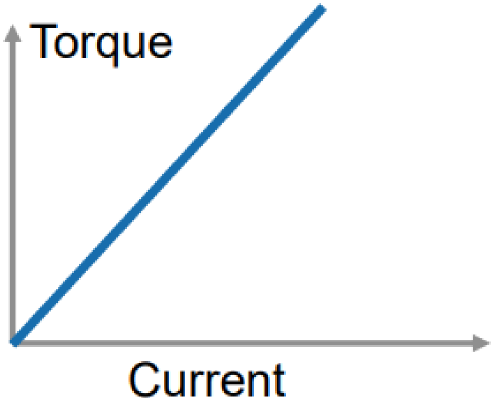
The block implements these equations, expressed in the motor flux reference frame (dq frame). All quantities in the motor reference frame are referred to the stator.

$$\omega_e = P\omega_m$$

$$\frac{d}{dt}i_d = \frac{1}{L_d}v_d - \frac{R}{L_d}i_d + \frac{L_q}{L_d}P\omega_m i_q$$

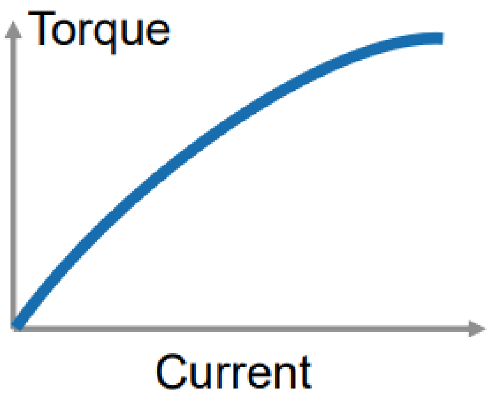
Model Fidelity

Plant Modeling



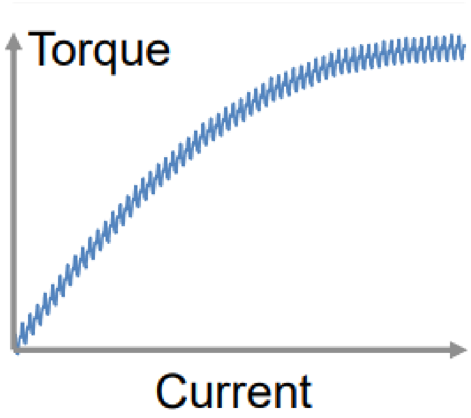
Linear Lumped Parameter

Motor Control Blockset
Simscape Electrical



Saturation

Simscape Electrical



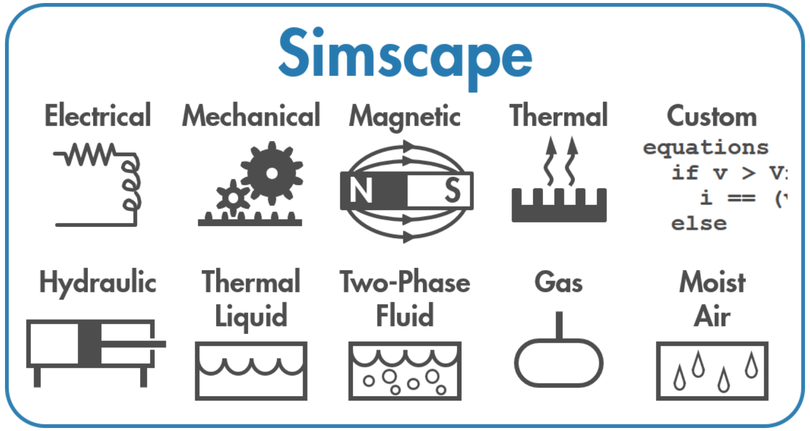
Saturation &
Spatial Harmonics

Simscape Electrical

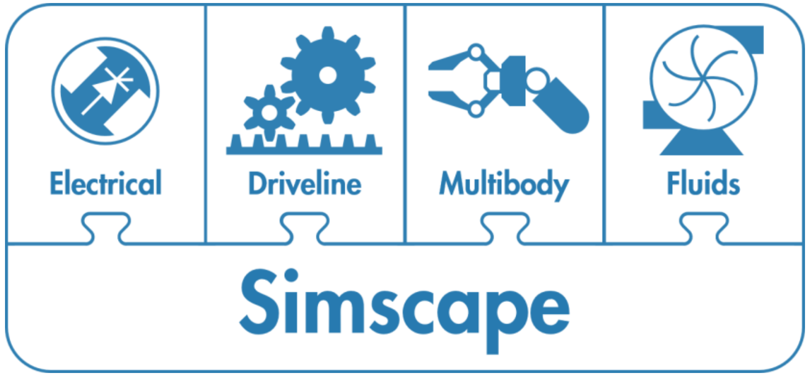
Simscape Products

Plant Modeling

- Simscape platform
  - Foundation libraries in many domains
  - Language for defining custom blocks
    - Extension of MATLAB
  - Simulation engine and custom diagnostics
- Simscape add-on libraries
  - Extend foundation domains with components, effects, parameterizations
  - Multibody simulation
  - Editing Mode permits use of add-ons with Simscape license only
  - Models can be converted to C code



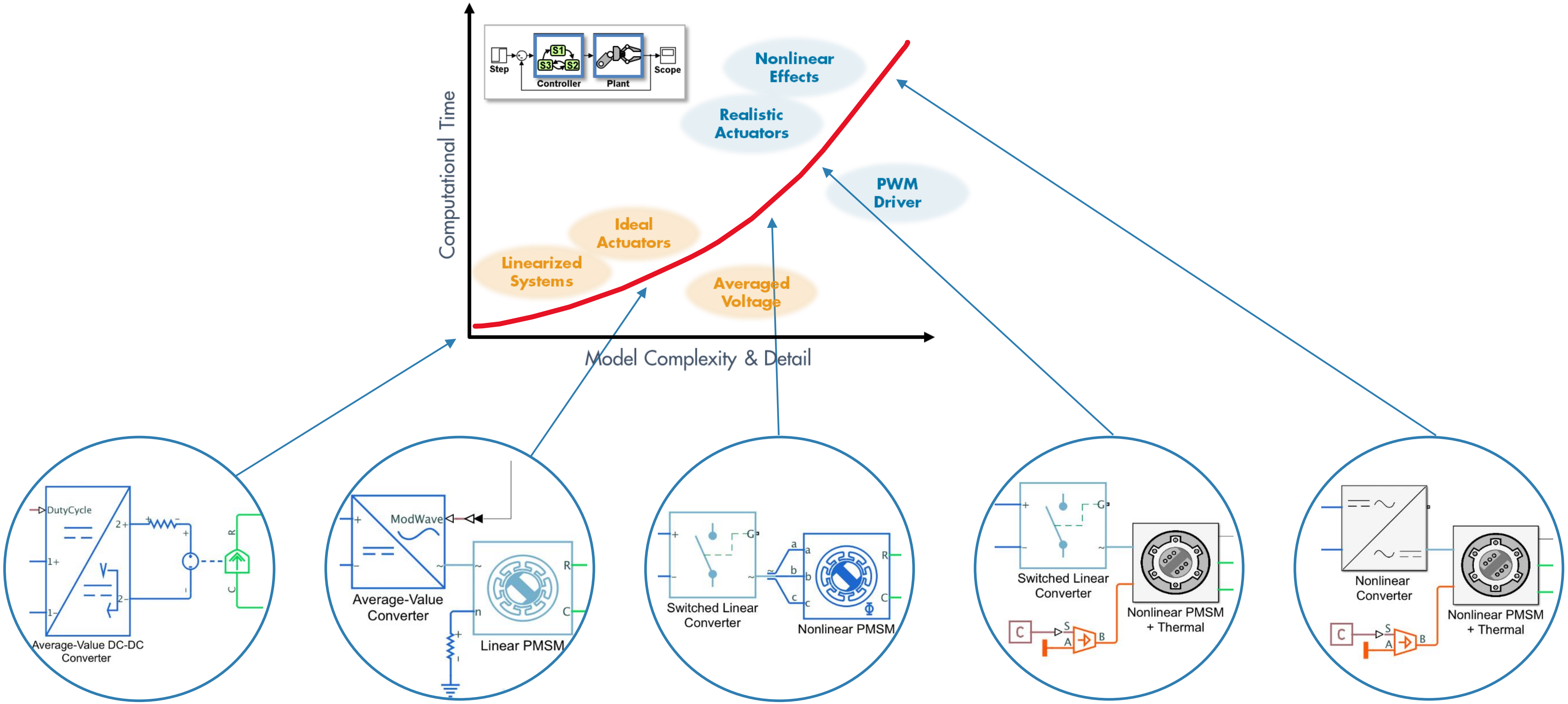
Simscape Foundation



Simscape Add-Ons

Trade Off - Balance Model Fidelity vs Simulation Speed

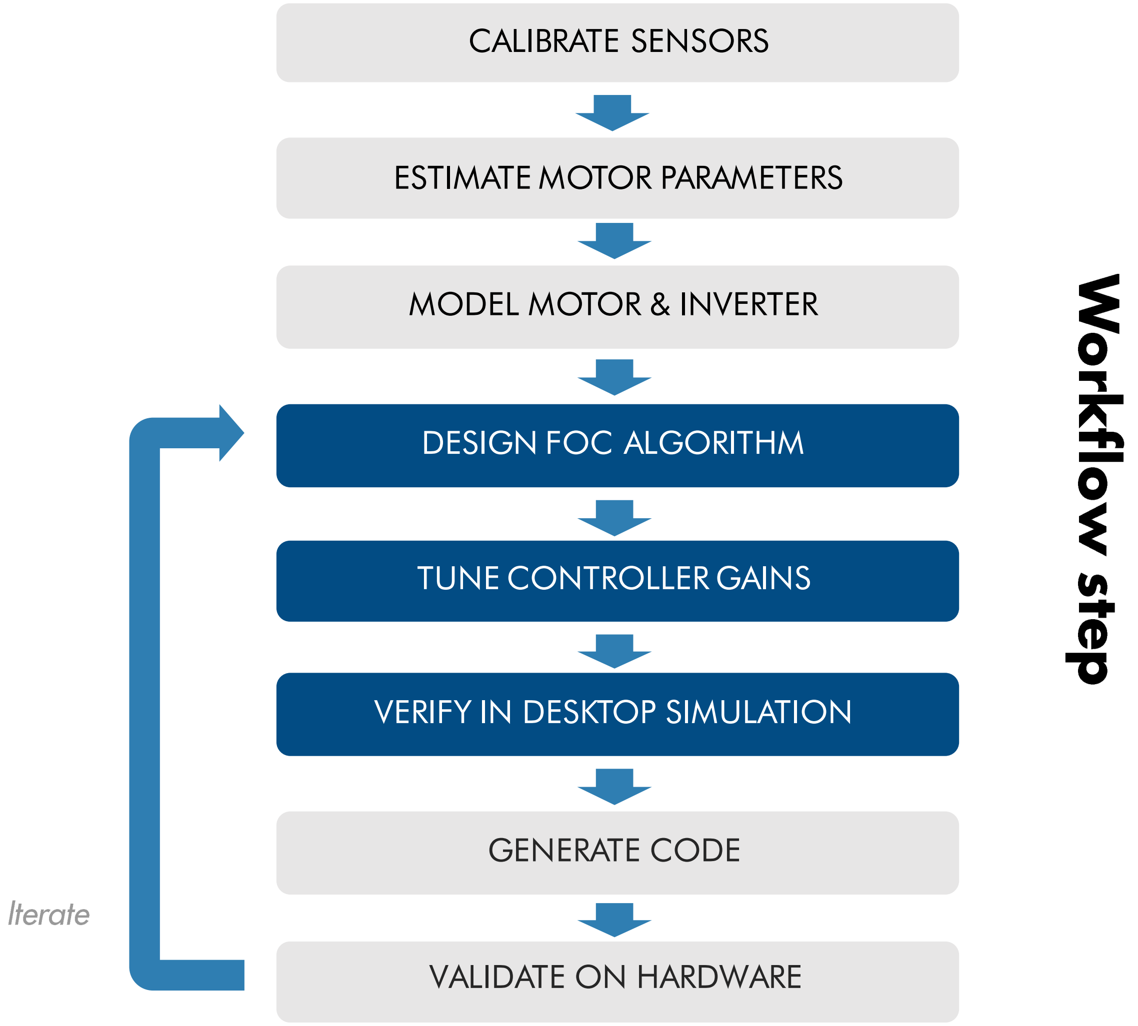
Plant Modeling



Agenda

From Desktop Simulation to Software Deployment

- Plant modeling
  - Sensors Calibration
  - Motor Parameters Estimation
  - Motor and Inverter Model
- Algorithm design with simulation
  - Field-Oriented control
  - Autotuning control gain
  - Verifying controller
- Software deployment
  - Code generation

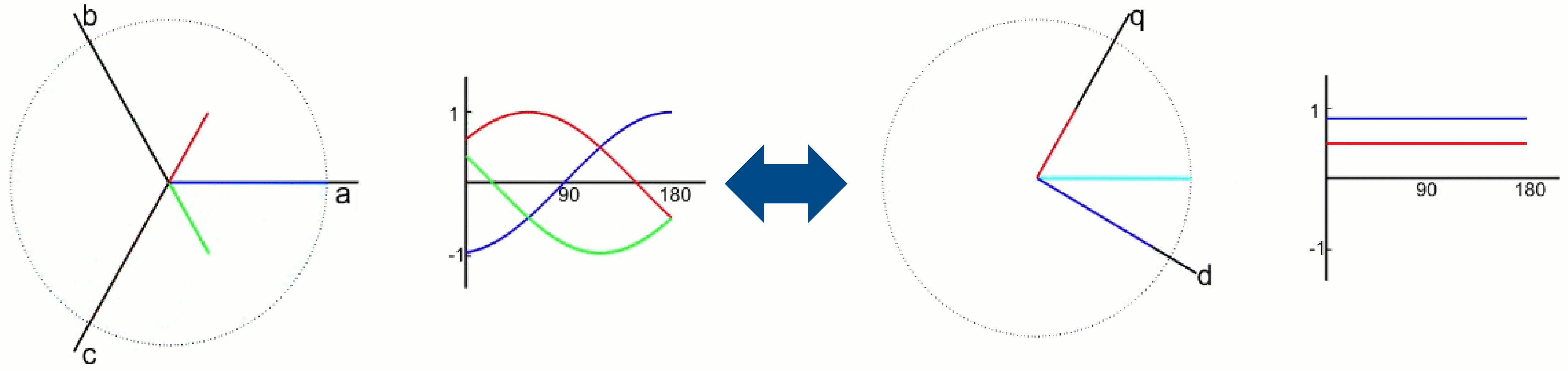


Modeling Field-Oriented Control (FOC)

A word about transforms

Measured current (A,B,C)
in time domain

Current control (d, q)

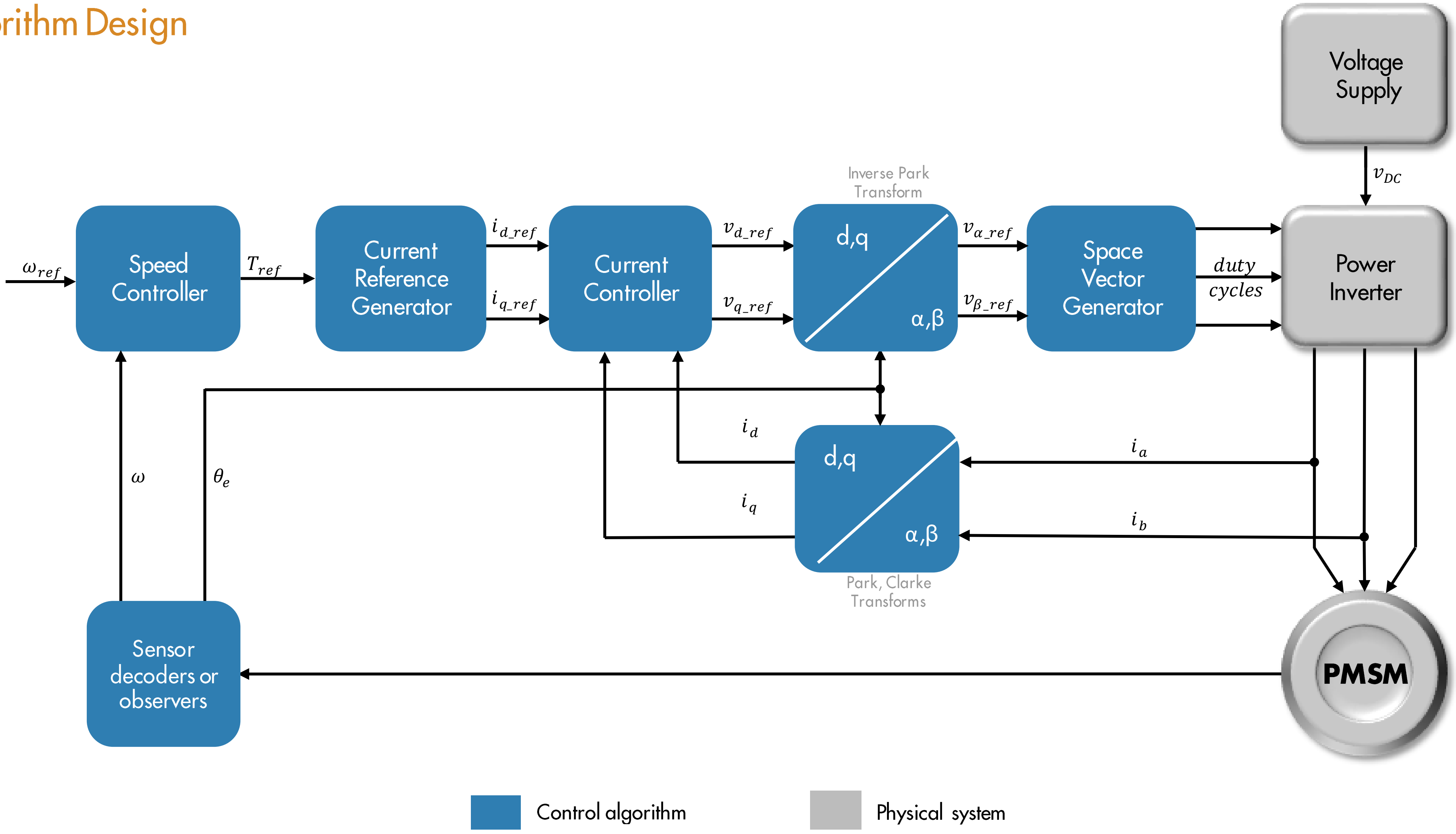


Clarke transform (abc ←→ $\alpha\beta$)

Park transform ($\alpha\beta$ ←→ dq)

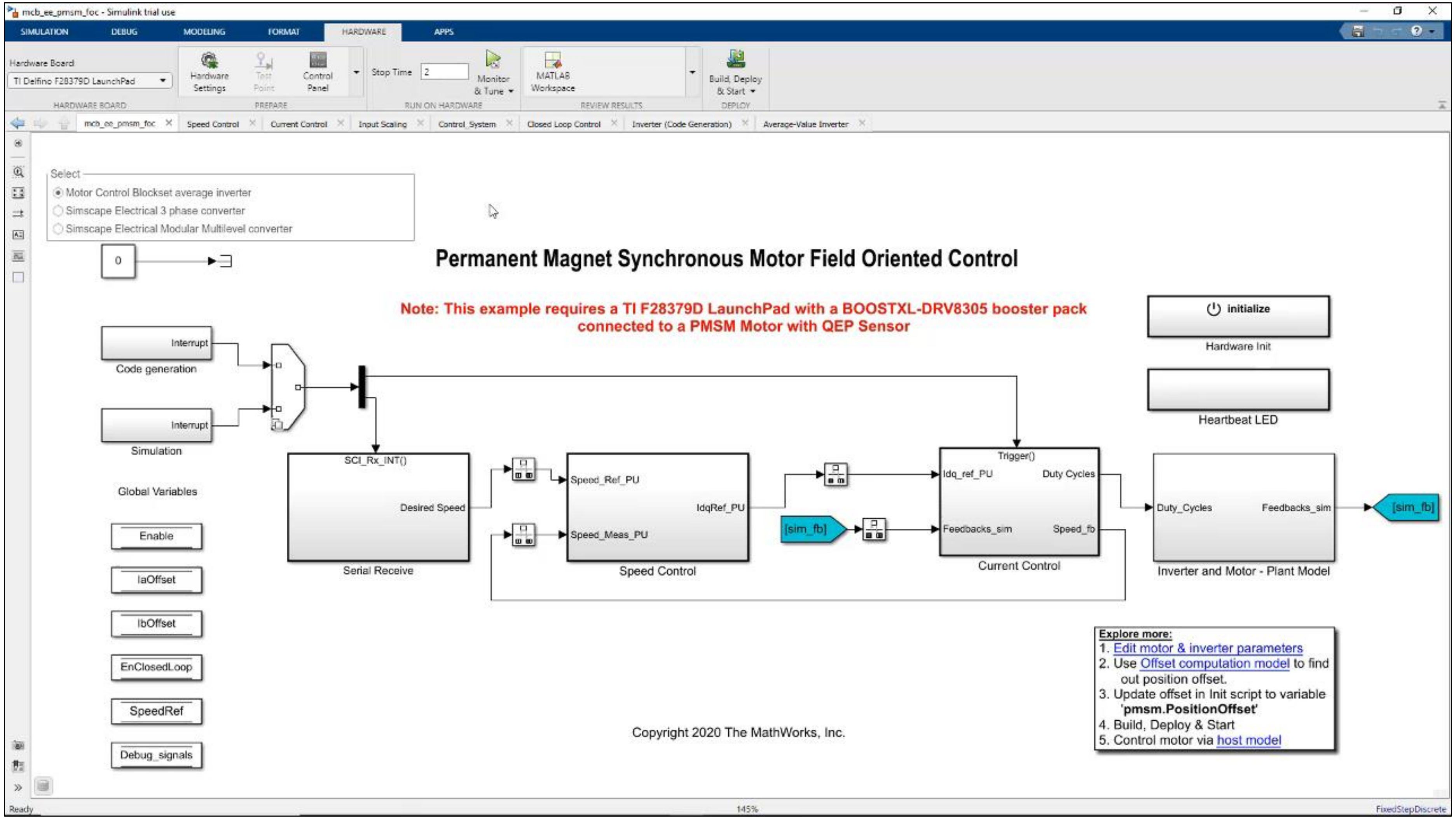
Modeling Field-Oriented Control (FOC)

Algorithm Design



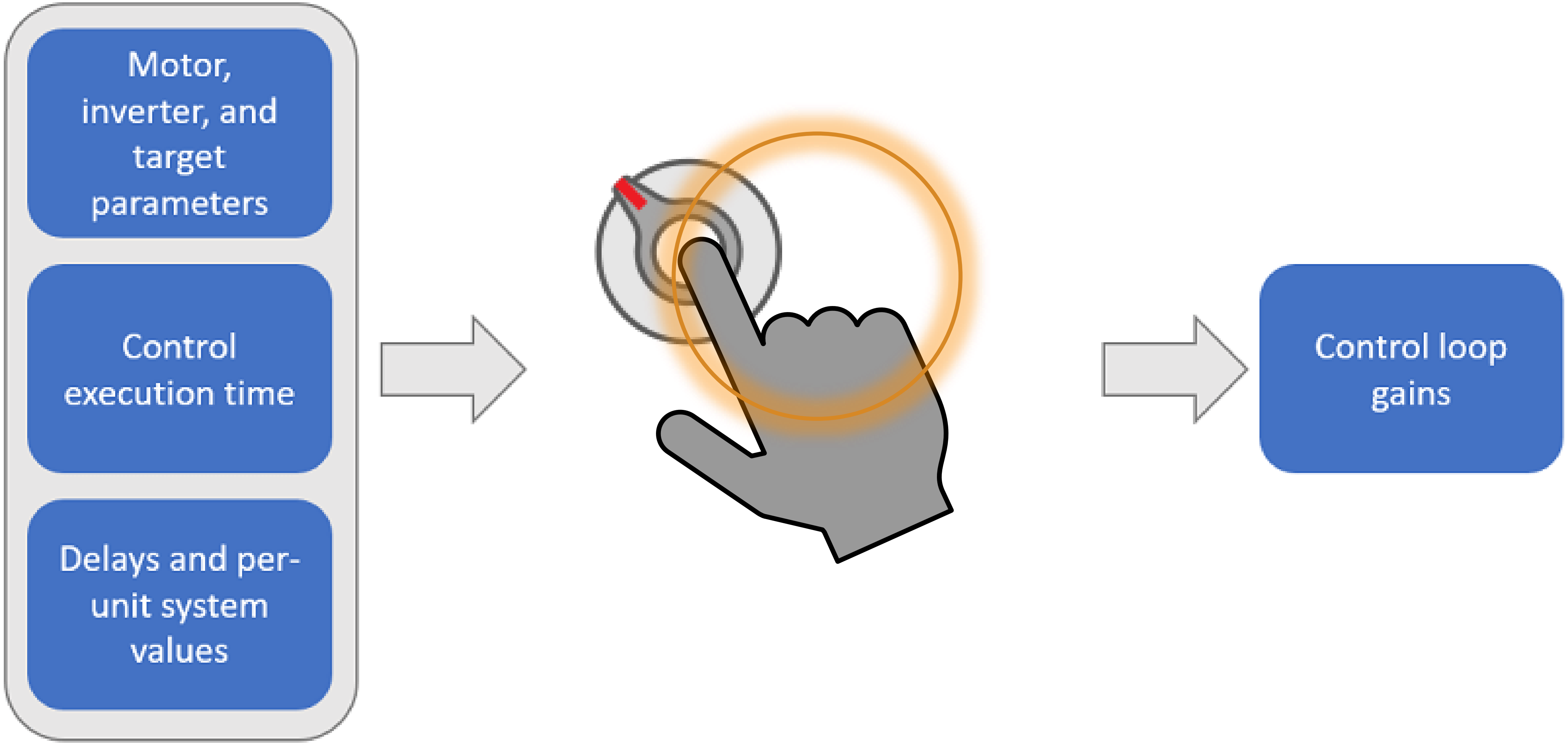
Modeling Field-Oriented Control (FOC)

Overview of the model



Autotuning controller gains

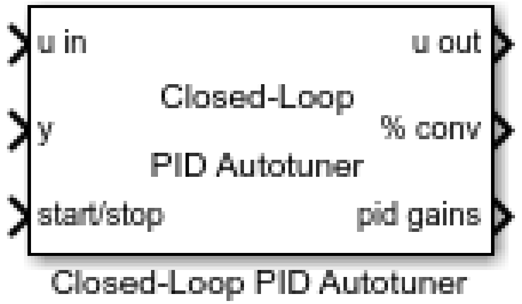
Multiple tuning methods available

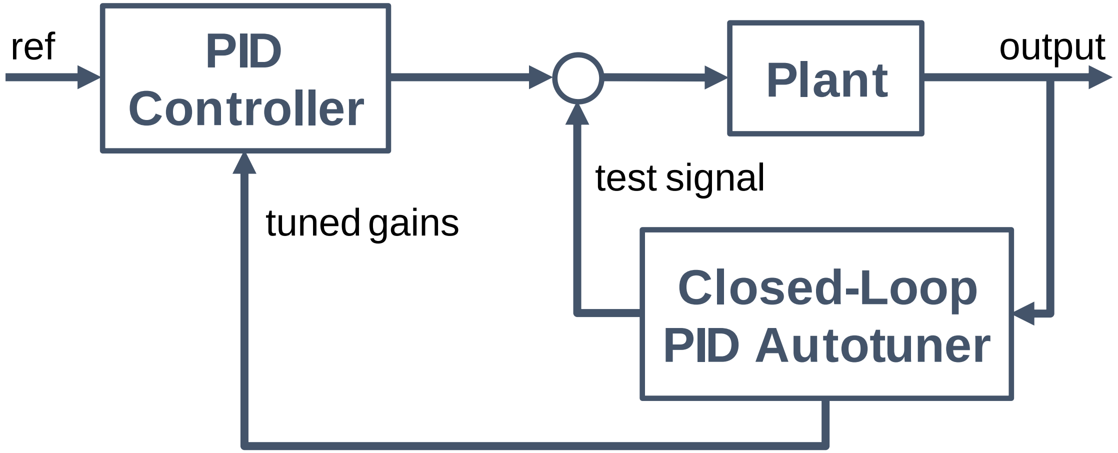


Autotuning controller gains

Closed-Loop PID Autotuner Block

- Algorithm details:
  - Injects a few superimposed sine waves, while maintaining closed-loop operation
  - Collects plant input-output data
  - Estimates frequency response in real-time
  - Tunes PID parameters to satisfy desired bandwidth and phase margin



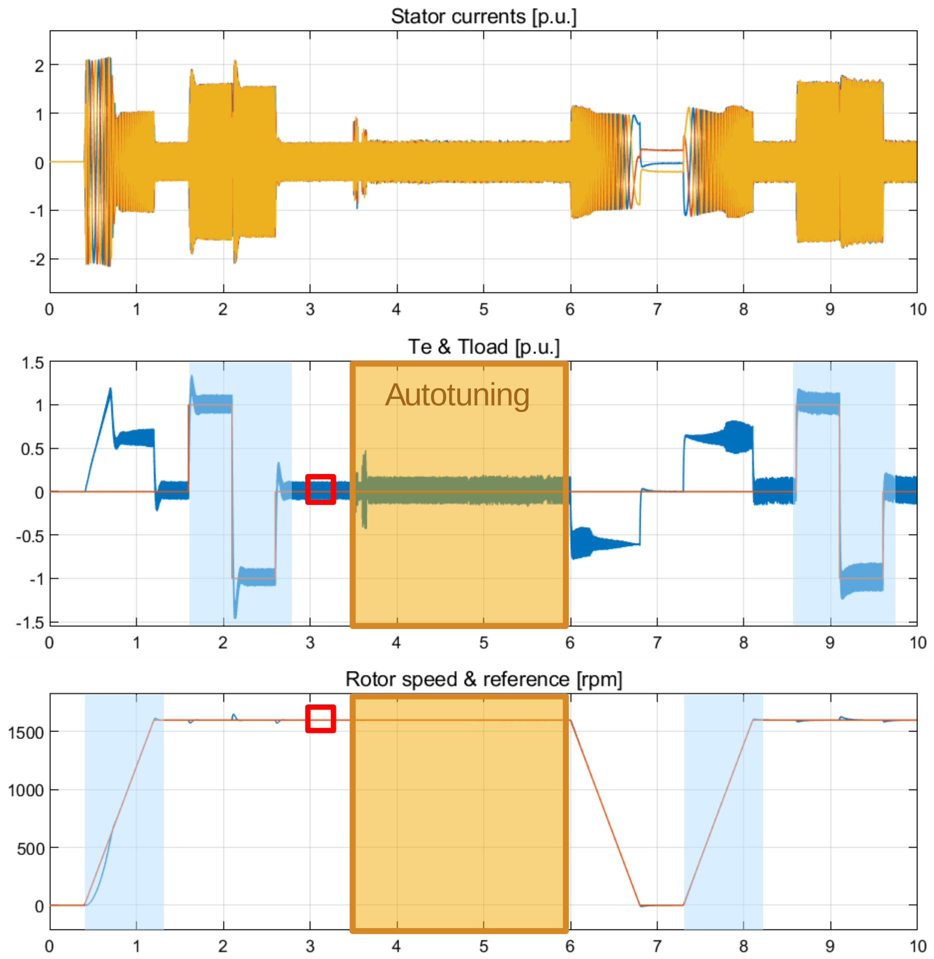


- Initial stable PID controller is required
- Option to deploy autotuning to embedded processor using Simulink Coder™

Requires Simulink Control Design™

Autotuning controller gains

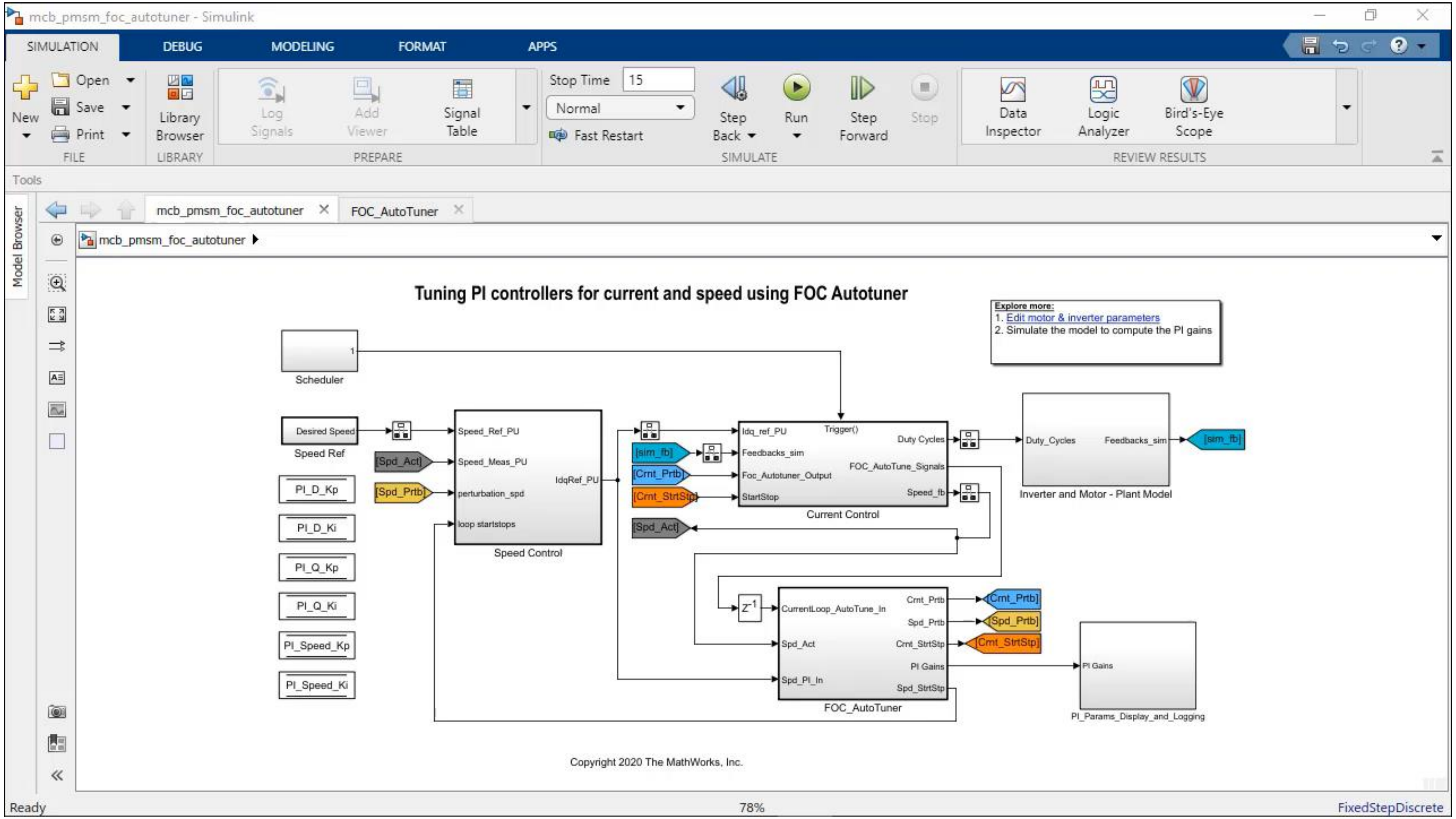
Tuning All Controller Gains in One Simulation



Motor speed is close to the nominal value while tuning

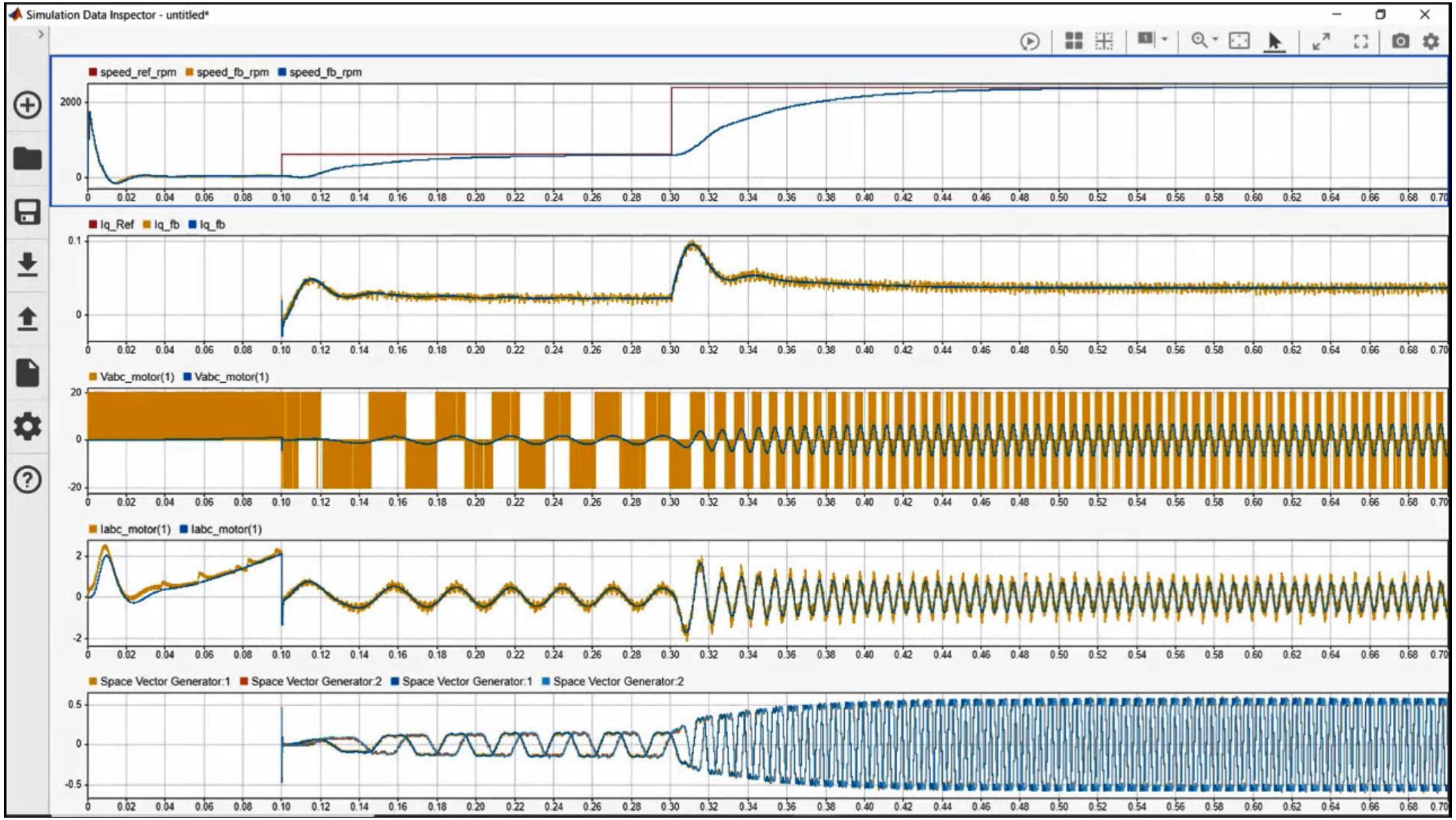
Autotuning controller gains

Overview of the workflow



Autotuning controller gains

Verify Controller Using Desktop Simulation



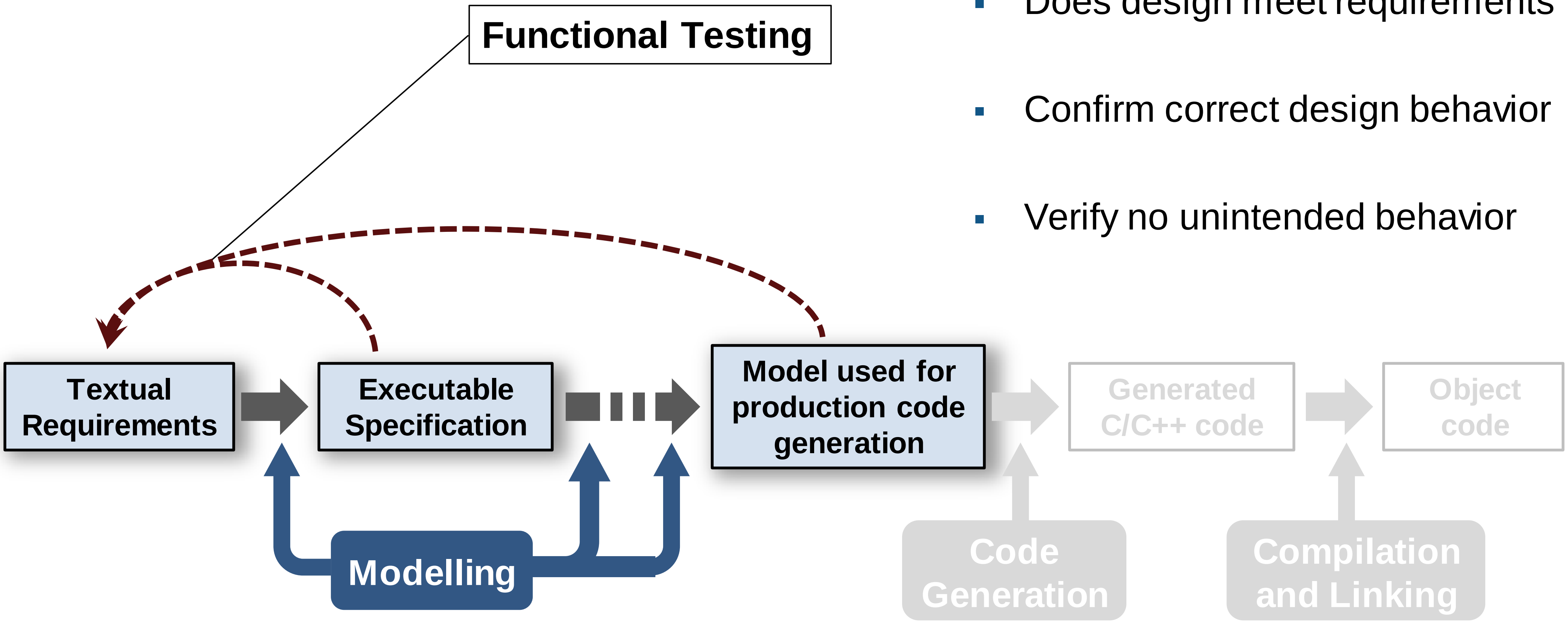
Verifying Controller

Key take-away

Simulation models are primary meant to support V&V activities

Verifying Controller

Overview



- Does design meet requirements
- Confirm correct design behavior
- Verify no unintended behavior

Verifying Controller

Functional Testing Process

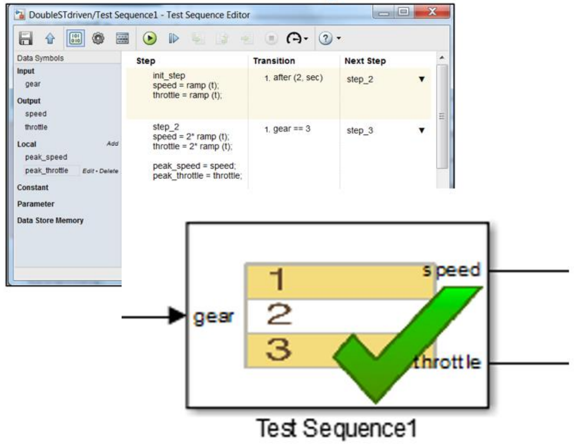
Author test-cases that are derived from requirements

- Use test harness to isolate component under test
- Test Sequence to create complex test scenarios

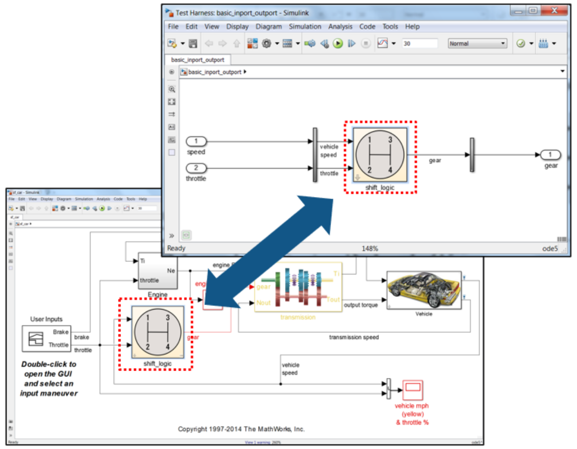
Manage tests, execution, results

- Re-use tests for regression
- Automate in Continuous Integration systems such as Jenkins

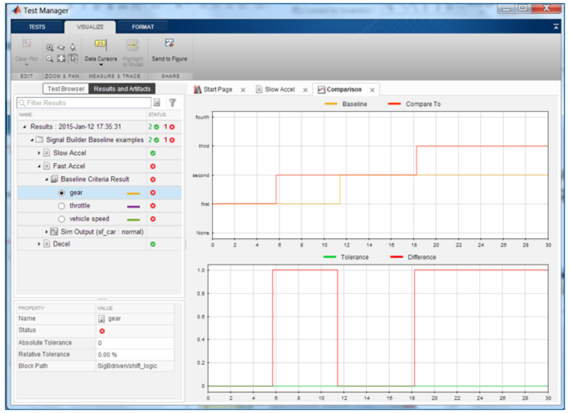
Test Sequence



Test Harness



Test Manager



Verifying Controller

V&V journey

Interactive testing

Reactive Testing

Coverage Analysis

Code Testing

Dynamic Testing

Edit-time checks

(Dead Logic Detection)
(Test Case Generation)

Requirement Proving

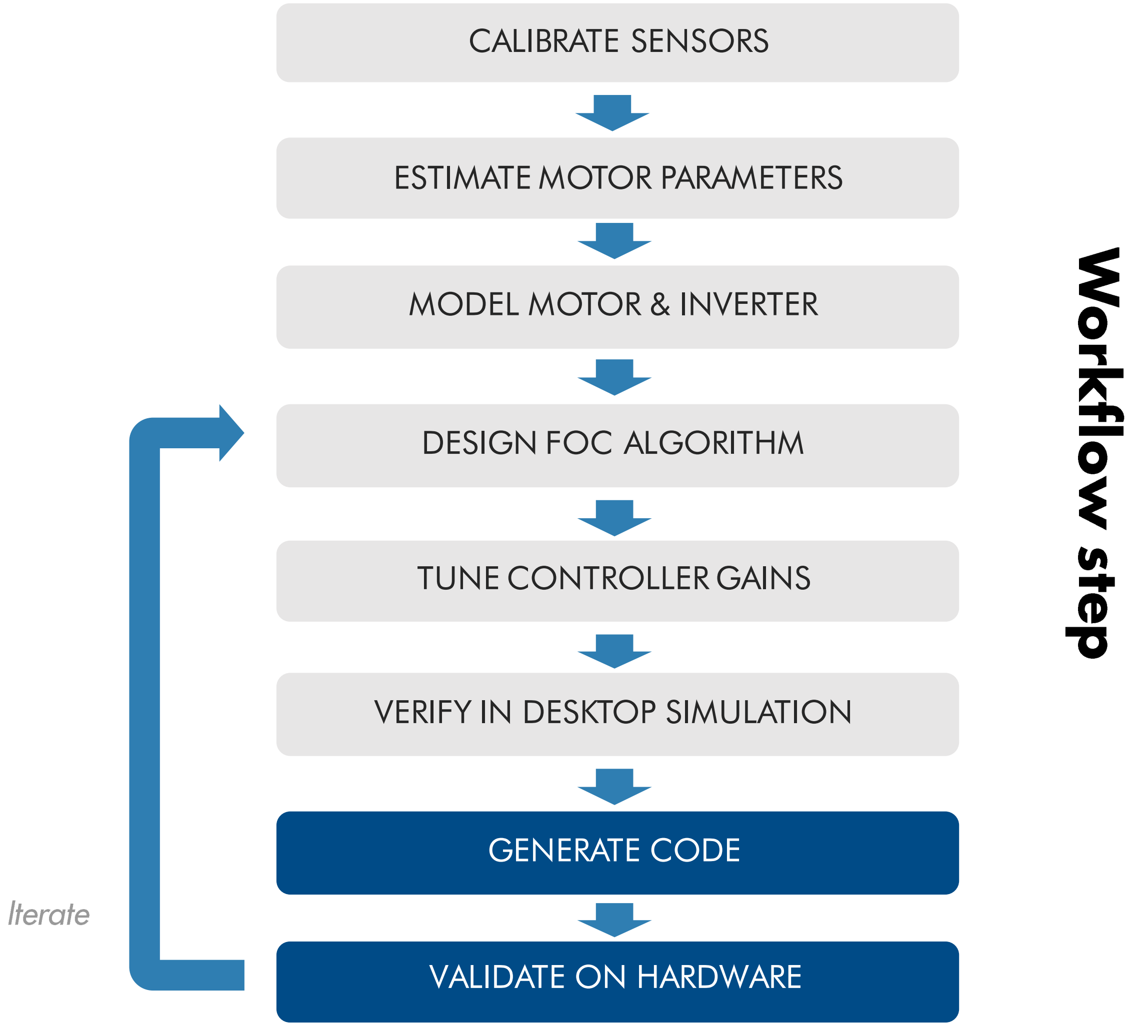
Code proving

Static Analysis

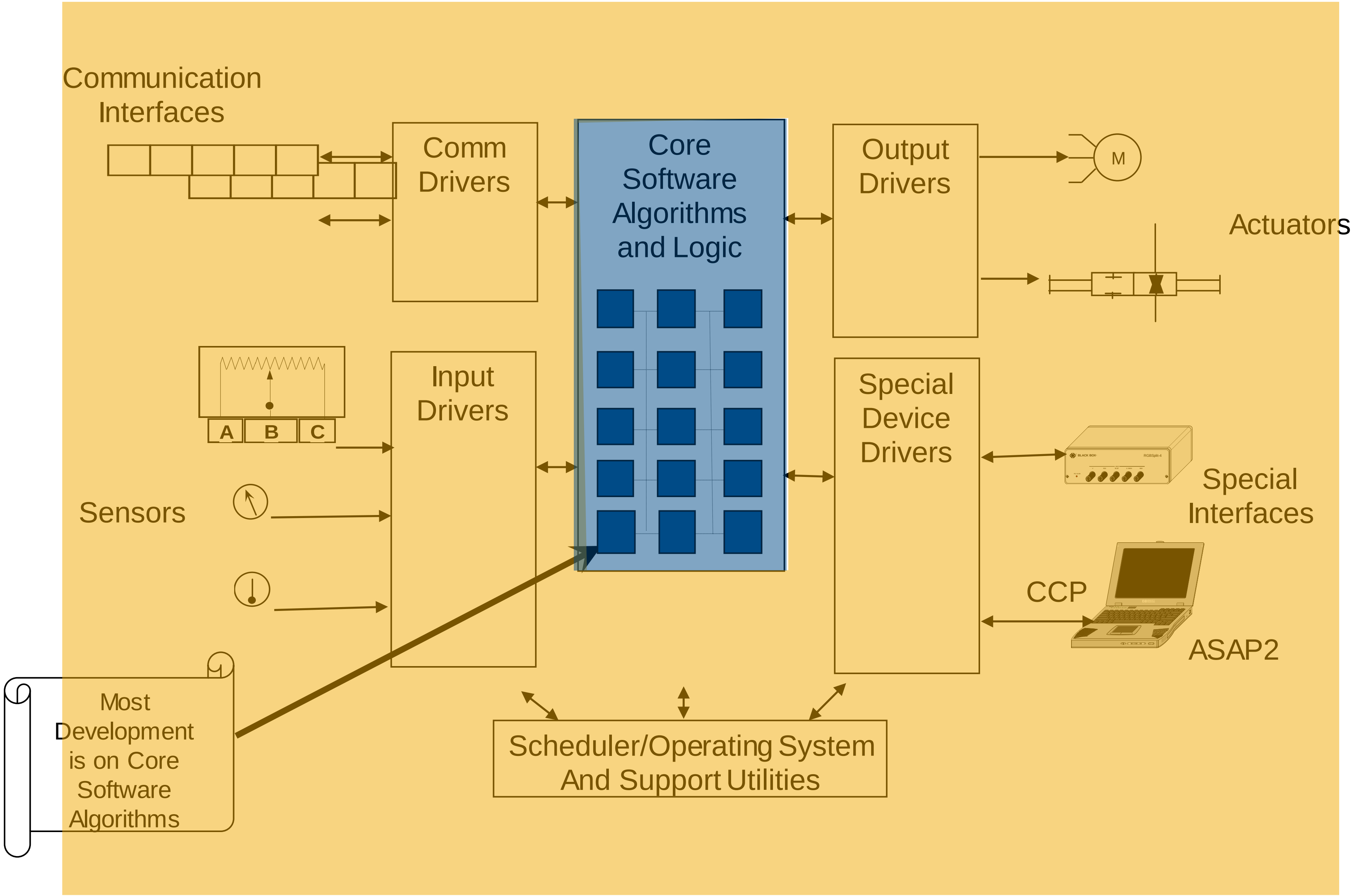
Agenda

From Desktop Simulation to Software Deployment

- Plant modeling
  - Sensors Calibration
  - Motor Parameters Estimation
  - Motor and Inverter Model
- Algorithm design with simulation
  - Field-Oriented control
  - Autotuning control gain
  - Verifying controller
- Software deployment
  - Code generation

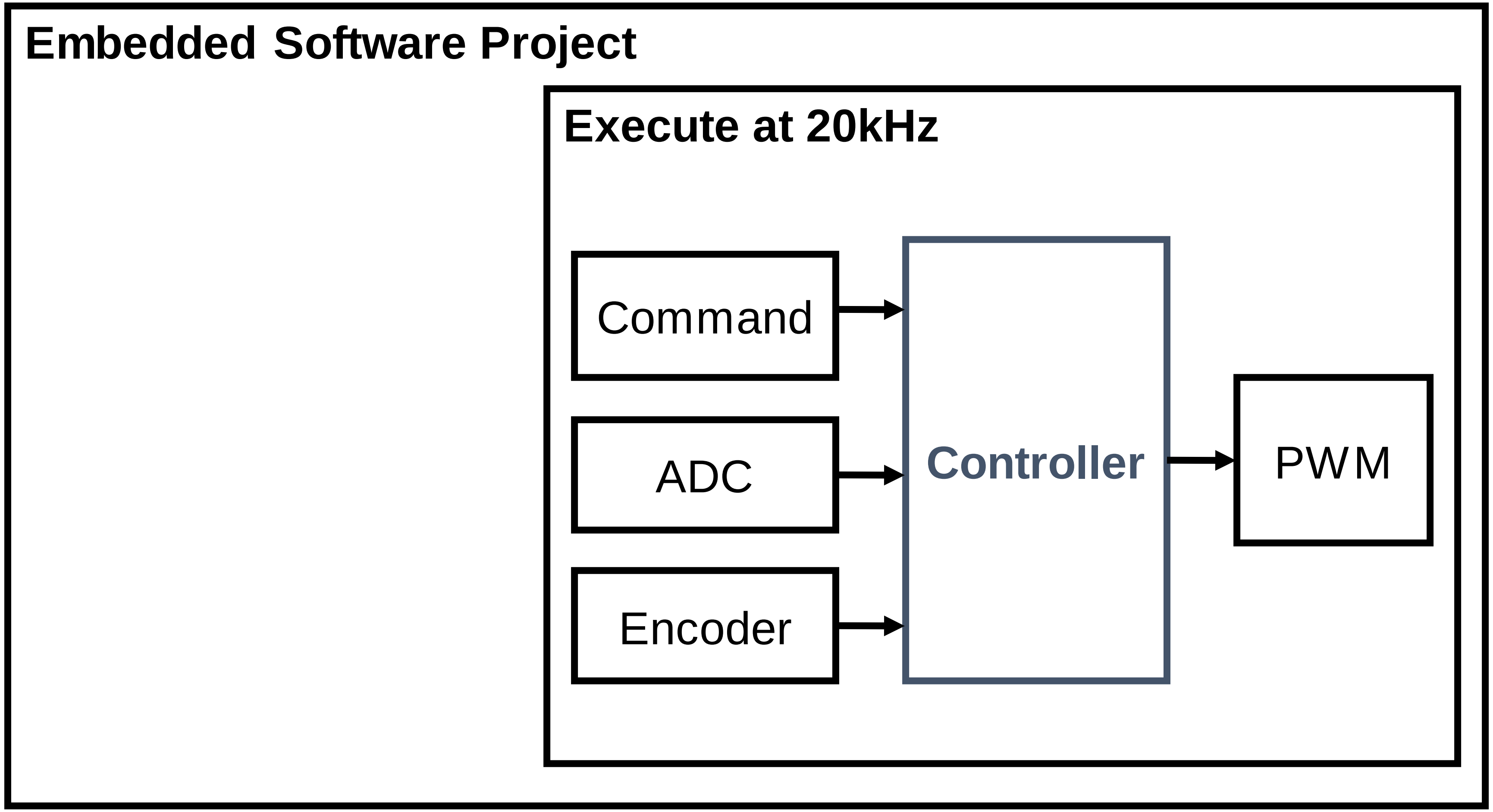


Simple Embedded Software Architecture



Integrating Generated Controller Code with an Embedded Software Project

Model Hand



Integrate Generated Controller Code with Your Hand-Coded Software Project

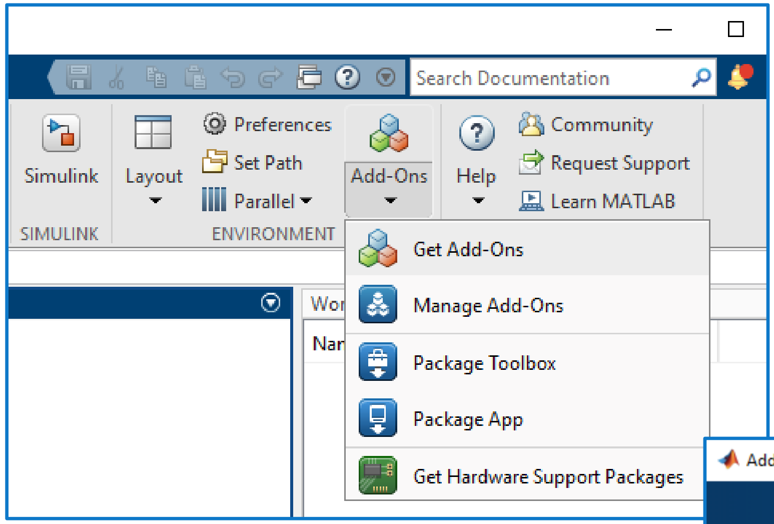
Model **Hand**

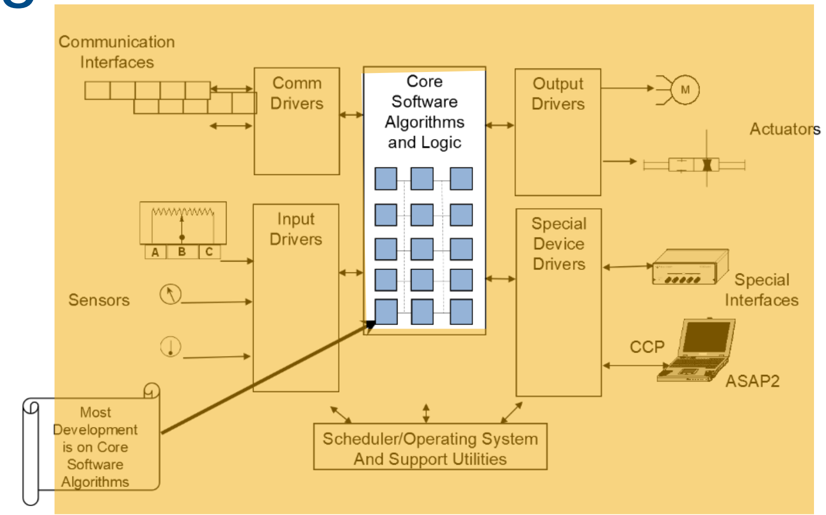
Embedded Software Project Pseudo-Code

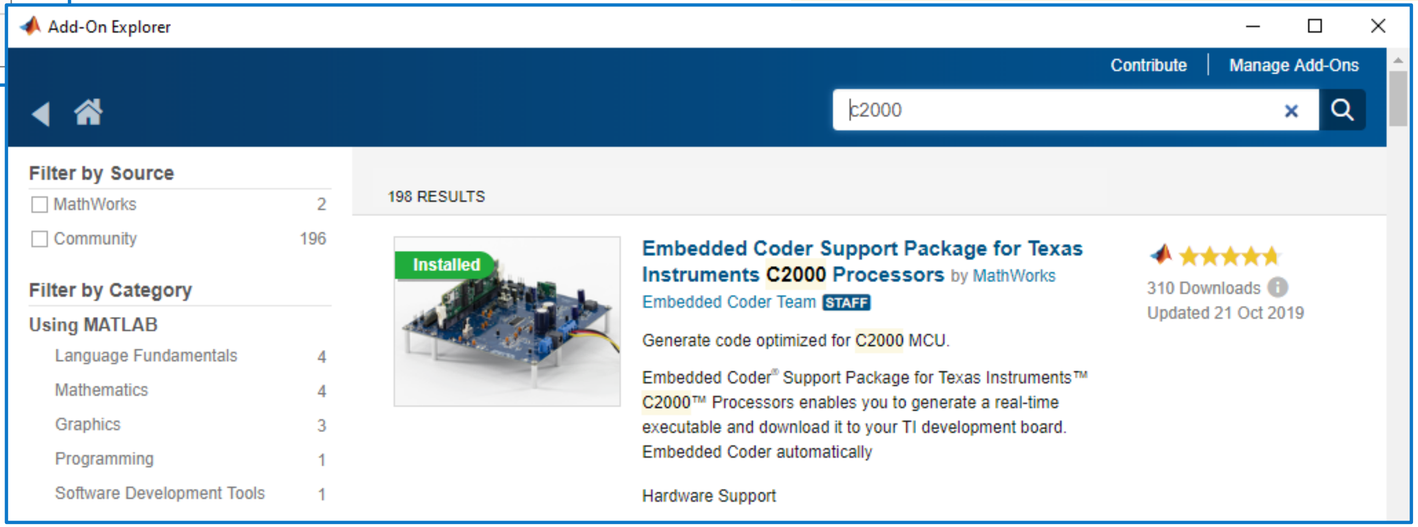
```
main()
{
    adcInit();
    encoderInit();
    pwmInit();

    controllerInit();

    while(1) {
    }
}
```

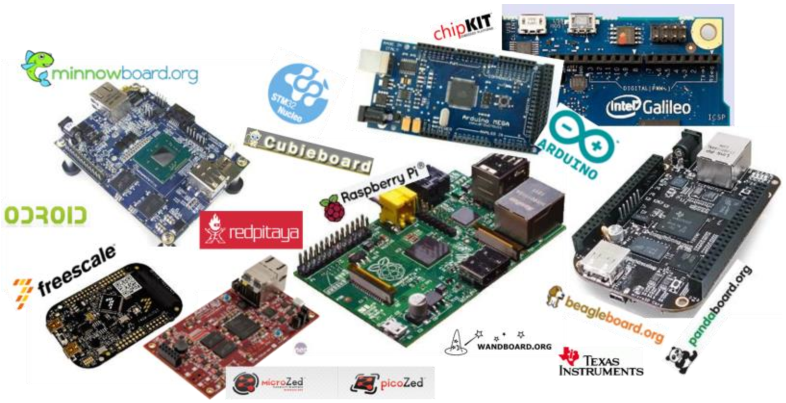
```
interruptServiceRountine()
{
    readAdcCountFromDriver();
    readEncoderCountFromDriver();

    controller();

    writePwmCountToDriver();
}
```

Embedded Coder Hardware Support Packages









Hardware Support Packages: https://www.mathworks.com/hardware-support/home.html

MathWorks TI C2000 Support Package for Embedded Coder

Supported devices:

- F2802x/3x/5x/6x/07x/004x
- F2833x/32x/37xS/37xD/38xS/38xD
- Fixed-point F280x/1x

F28379D LaunchPad

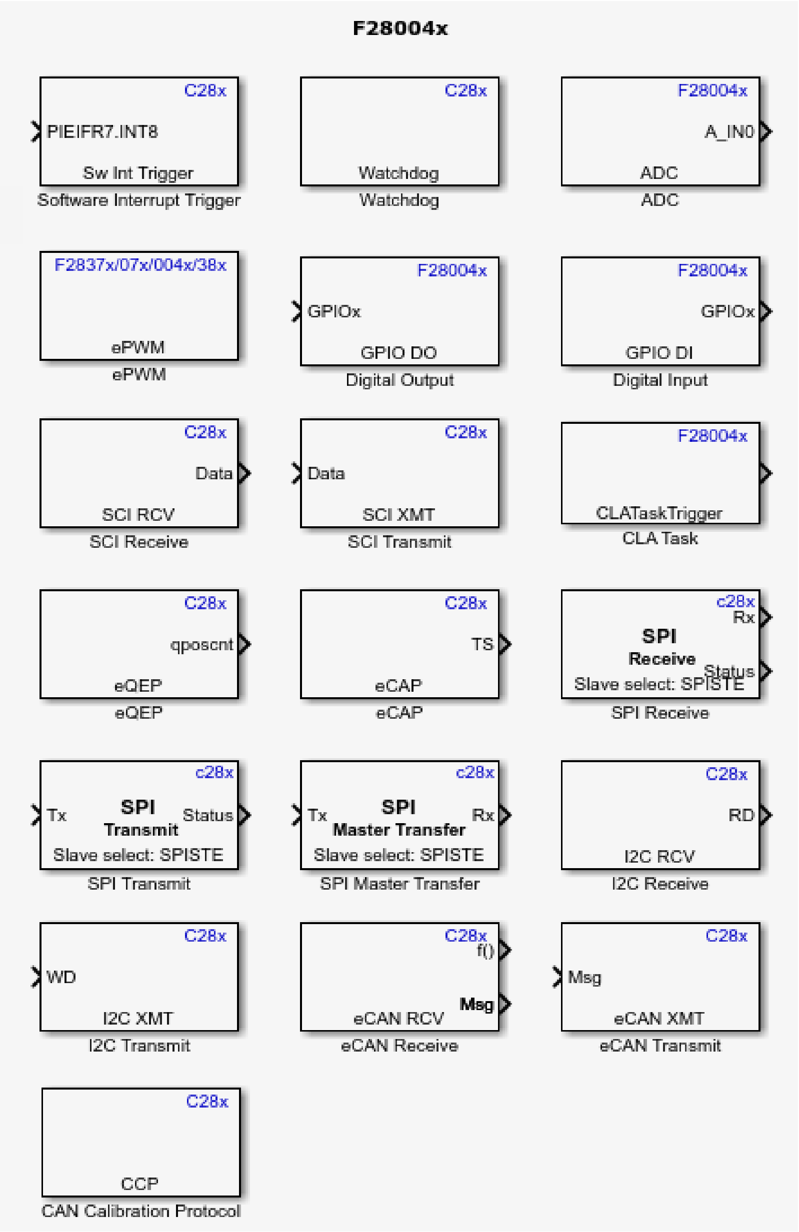
Scheduling the generated code:

- Periodic tasks
- Idle tasks
- Interrupts (Hardware, Software)
- Advanced concepts:
  - Pre-emptive rate-monotonic scheduler
  - Base rate interrupt replacement
  - Peripheral triggers (launch A/D conversion from PWM)
  - Running on the CLA
  - Loading in Flash, running in RAM
  - Using DMA

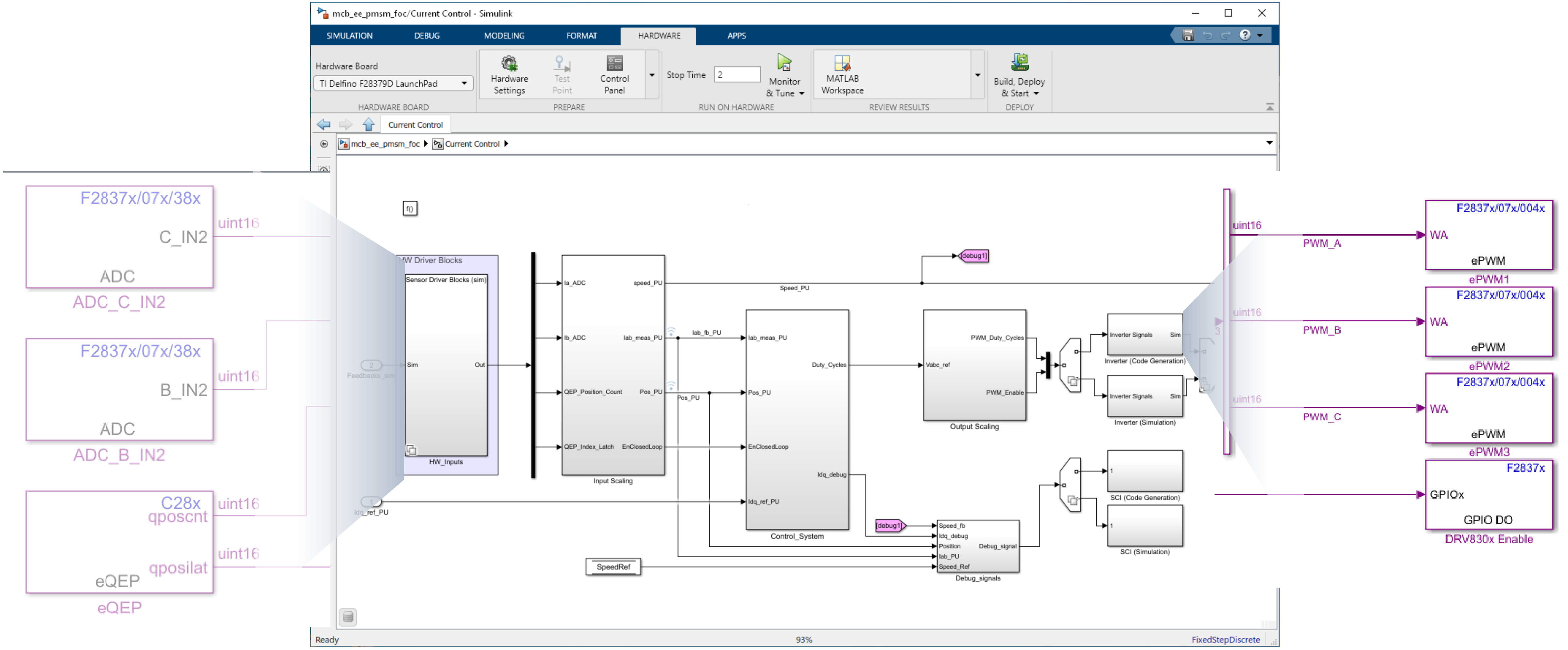
Supported TI C2000 drivers

- ADC, AIO, Comparator,
- GPIO, eQEP, ePWM, eCAP,
- eCAN, I2C, SCI, SPI, LIN
- Watchdog, DMA

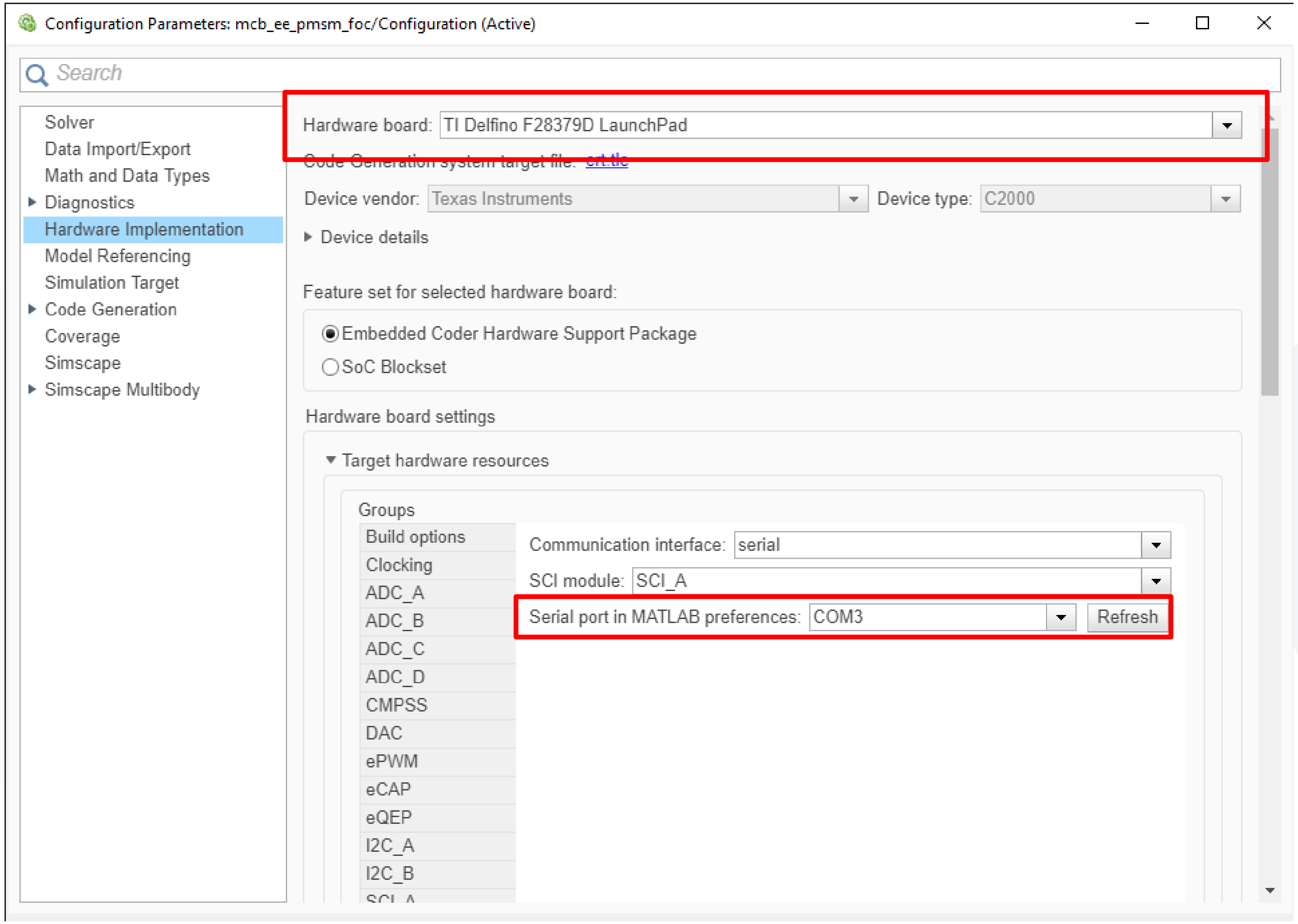
- Motor control position sensing
  - Optical encoder (using eQEP)
  - Hall sensors (using eCAP)
  - Sensorless (using SMO)



Prepare the Model for Code Generation Using Supported TI C2000 Drivers Blocks

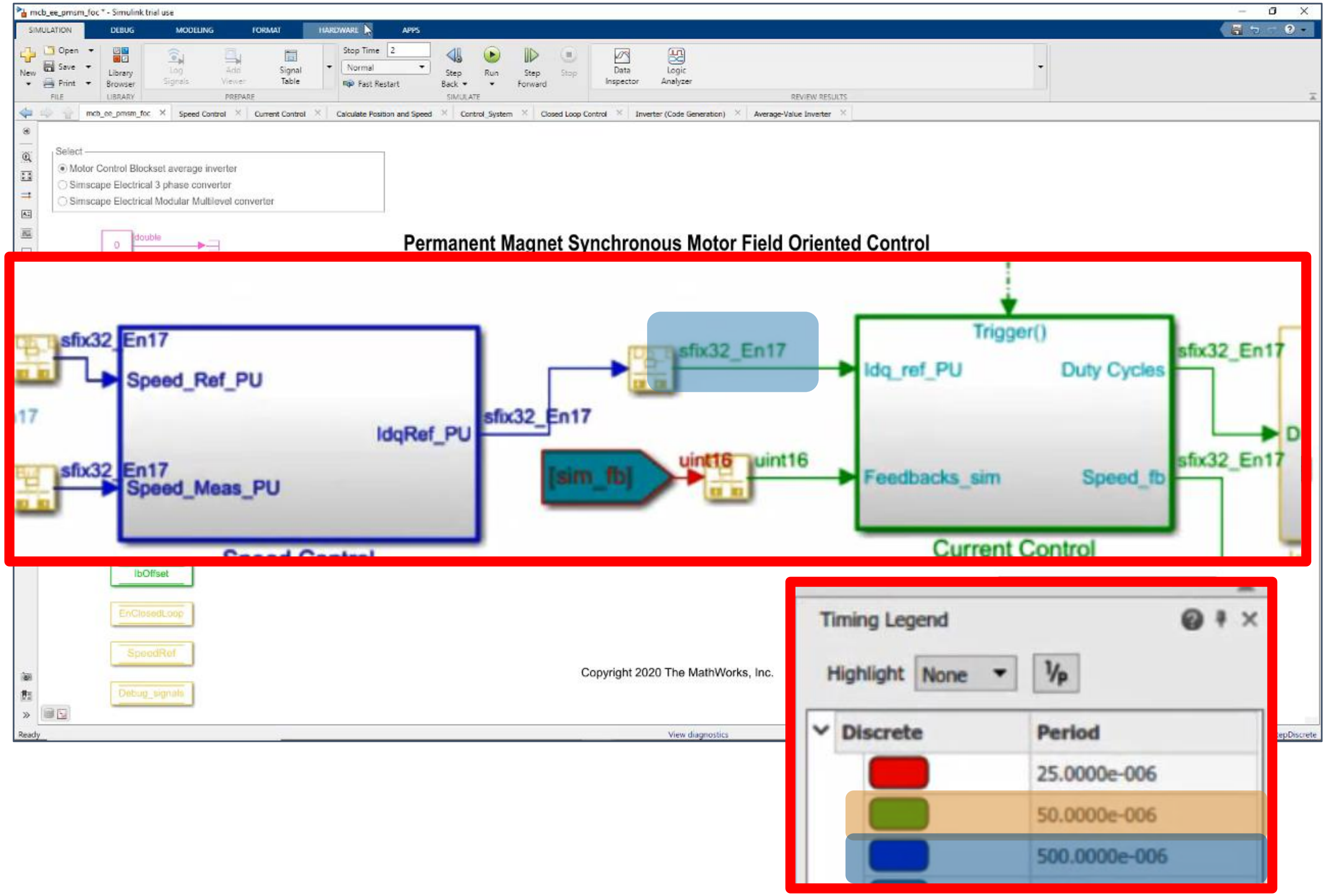


Prepare the Model for Code Generation Using Supported TI C2000 Drivers Blocks

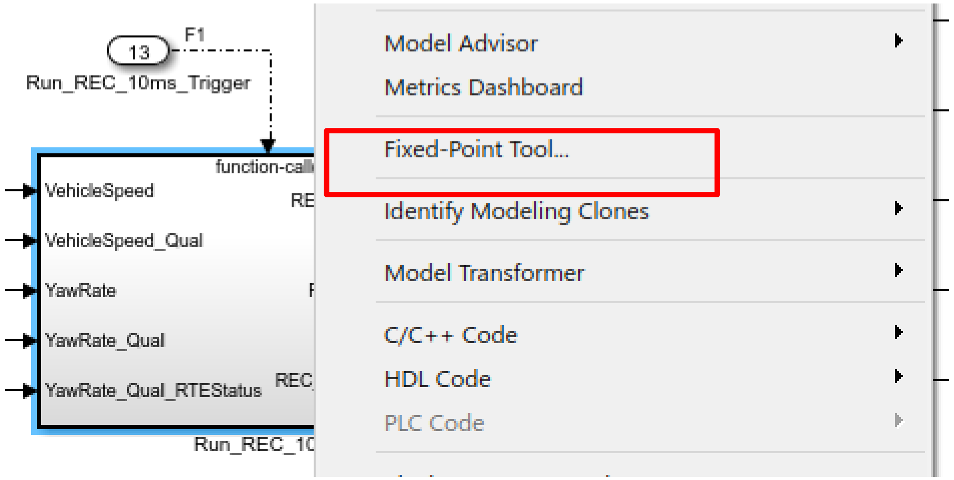


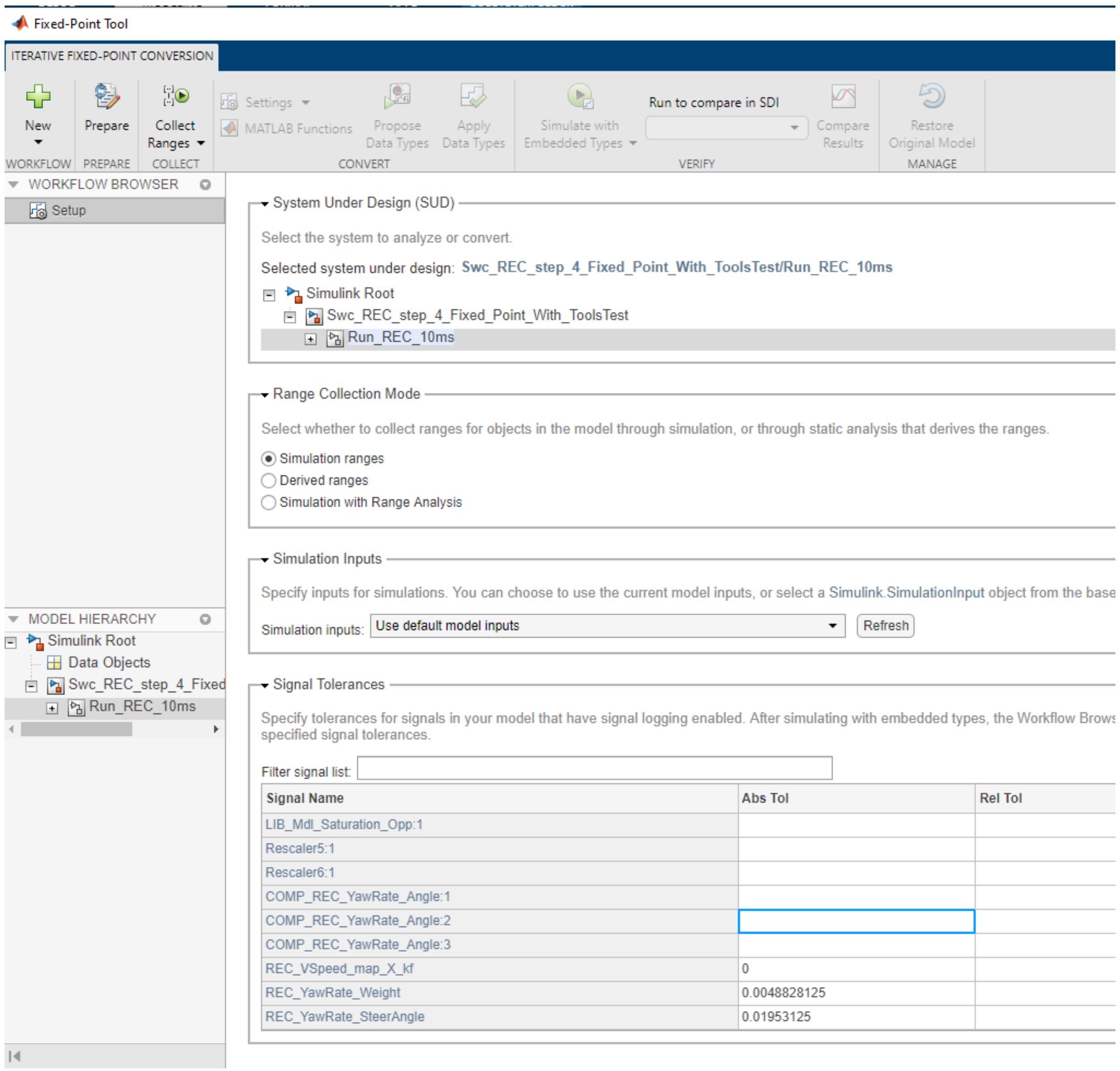
Deployment on the Target

- Generate code (floating and fixed-point)
- Use host model to control and debug
- Validate on hardware



Fixed-Point conversion

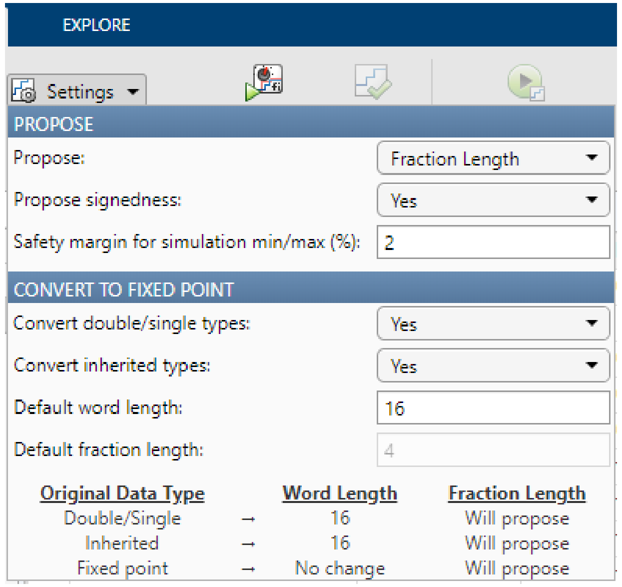


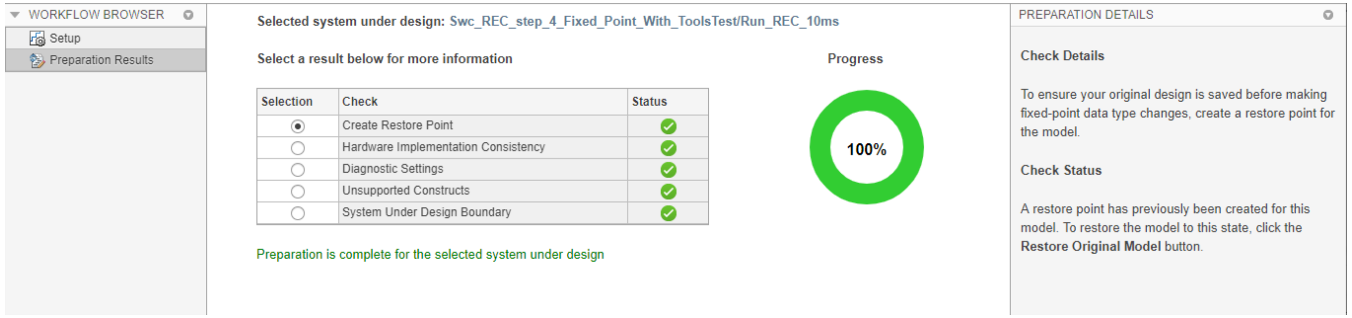


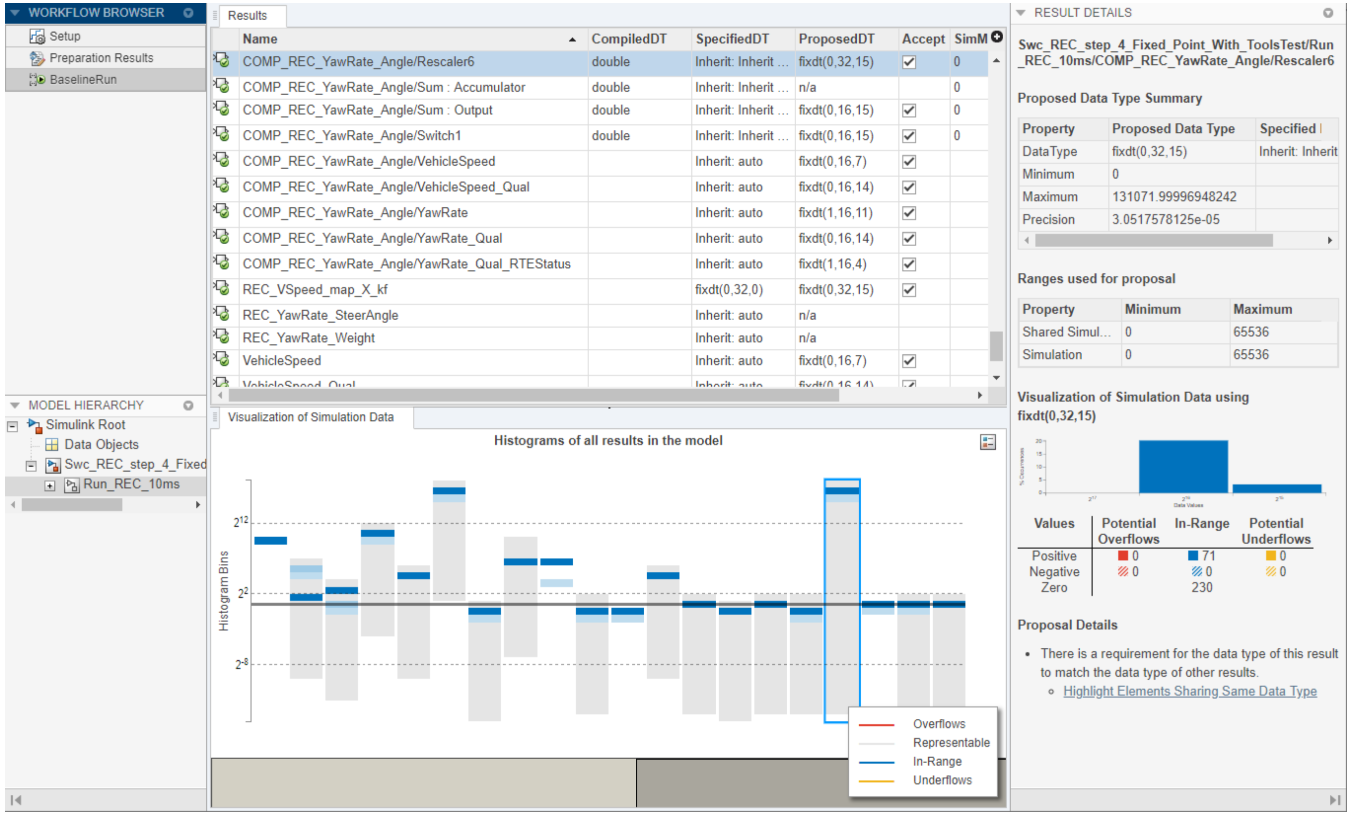
- Run the tool on the system to convert
- Chose your conversion method

Fixed-Point conversion

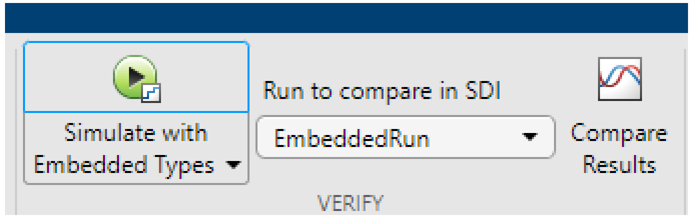
- Prepare the environment
- Configure your options
- Accept or modify the datatype proposition





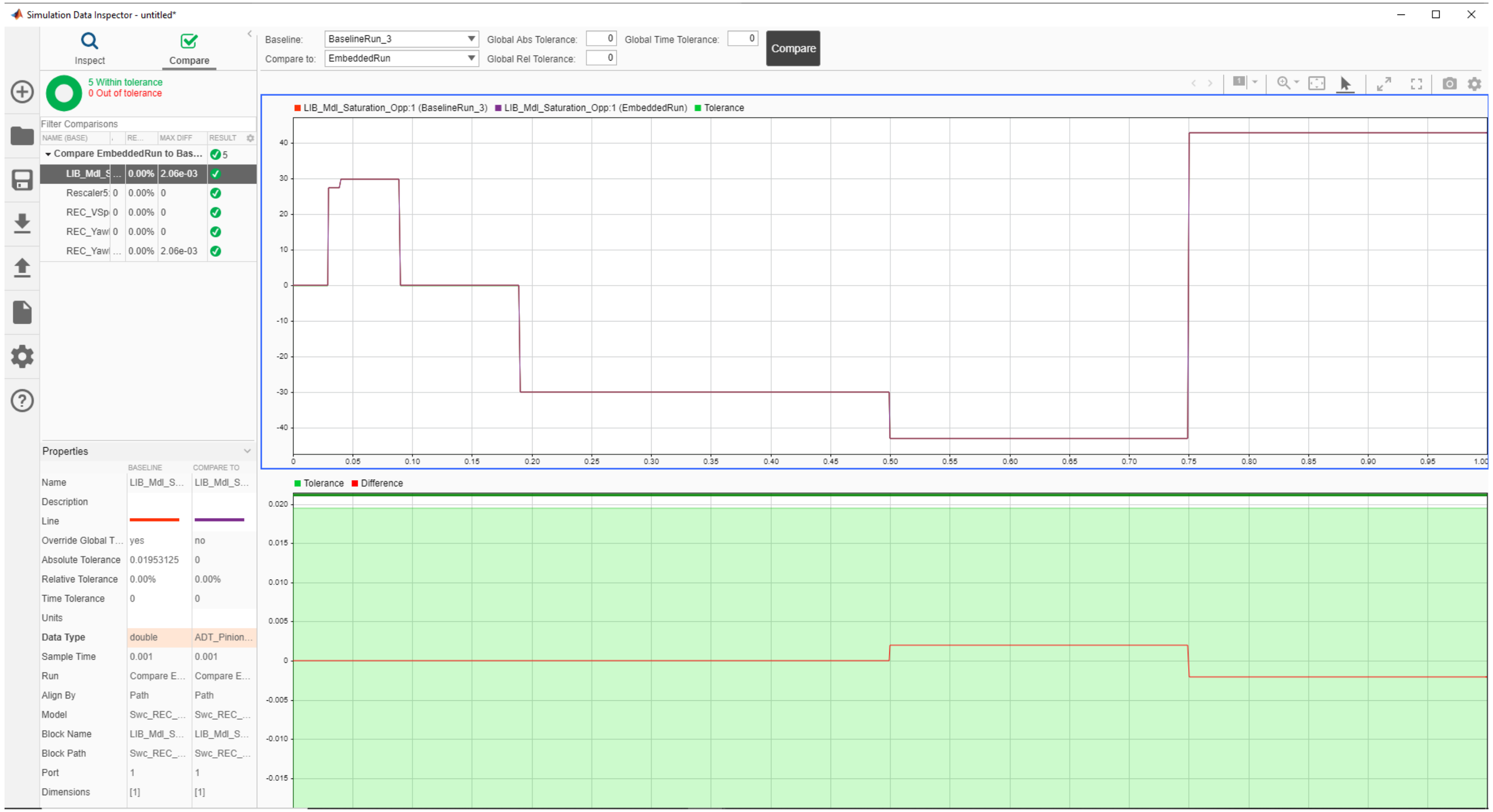


Fixed-Point conversion



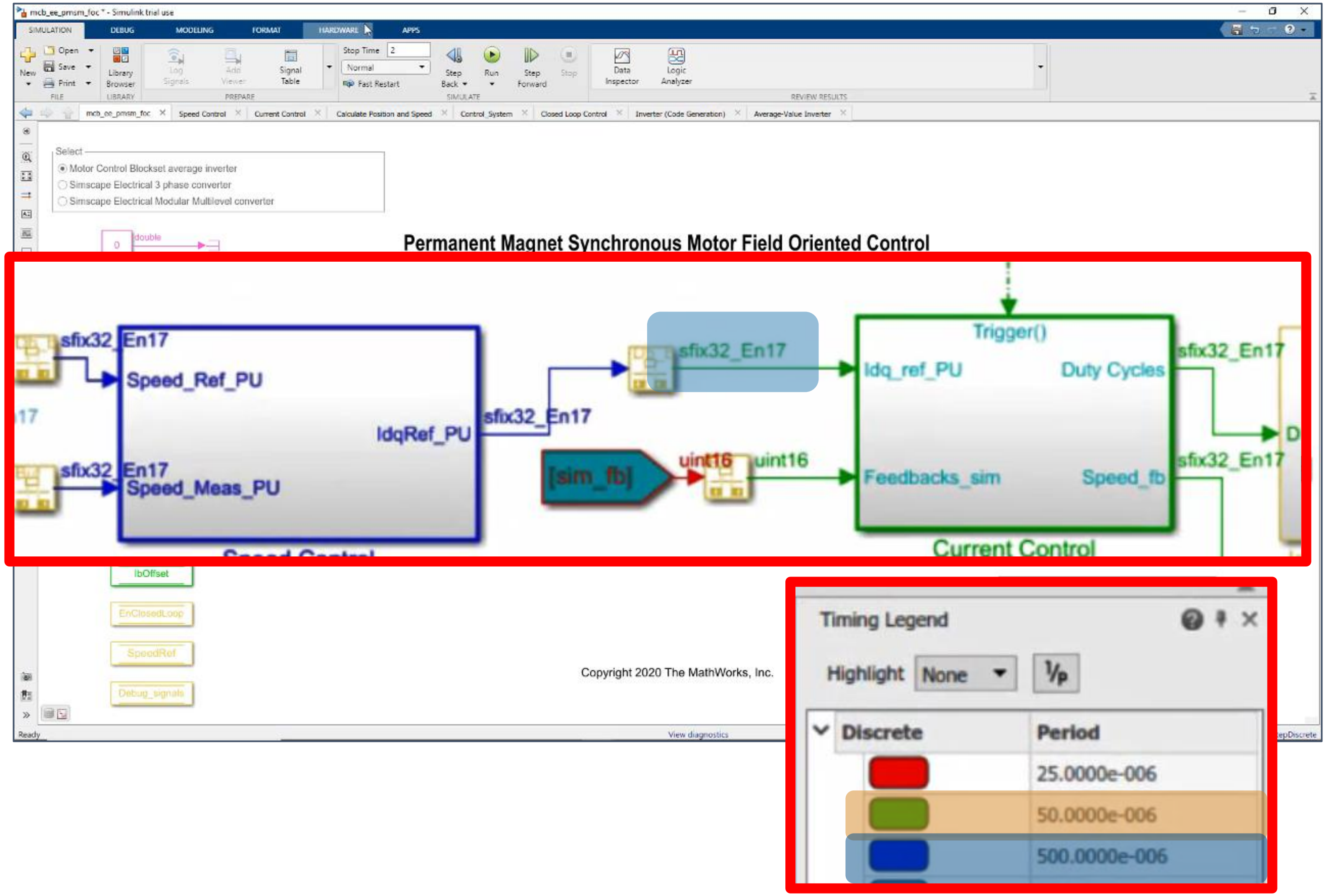
- Run again with your new datatype

- Compare automatically with floating point results



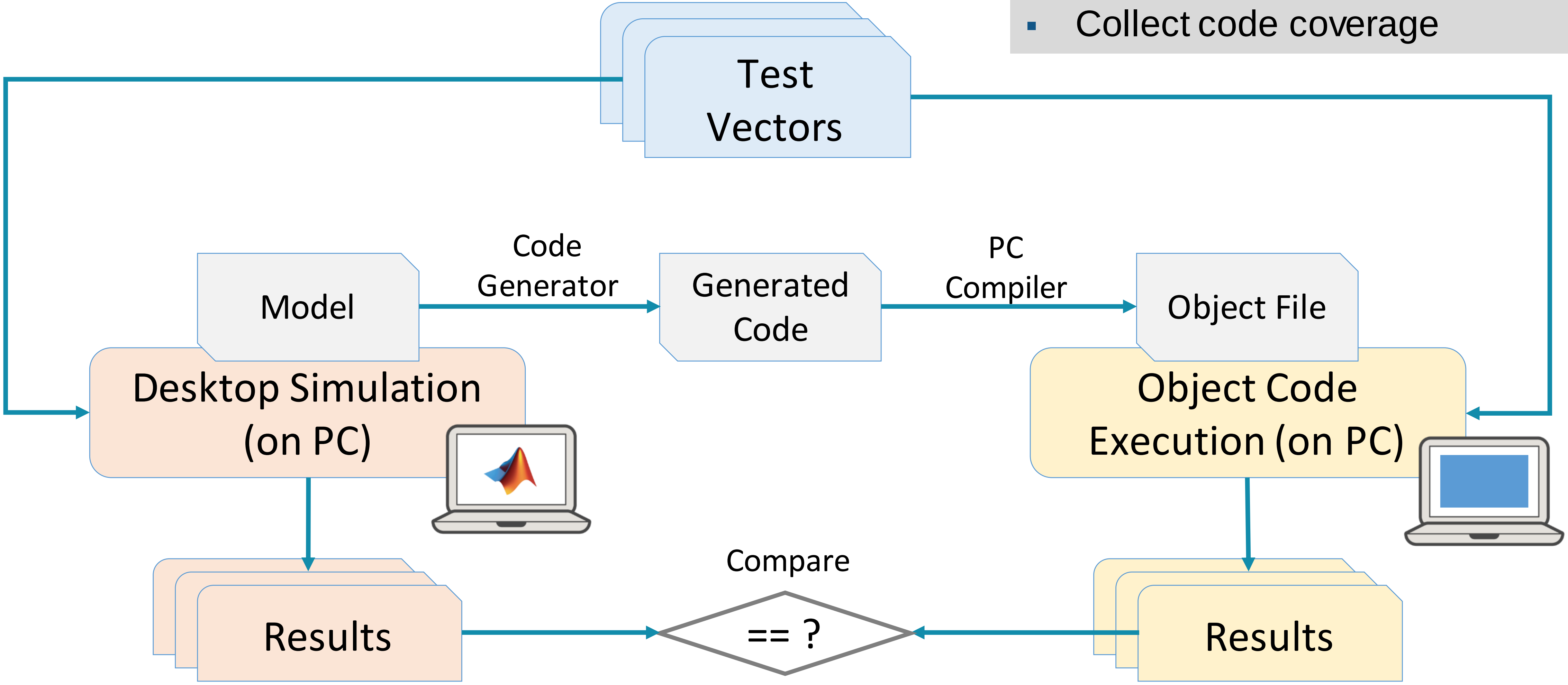
Deployment on the Target

- Generate code (floating and fixed-point)
- Use host model to control and debug
- Validate on hardware

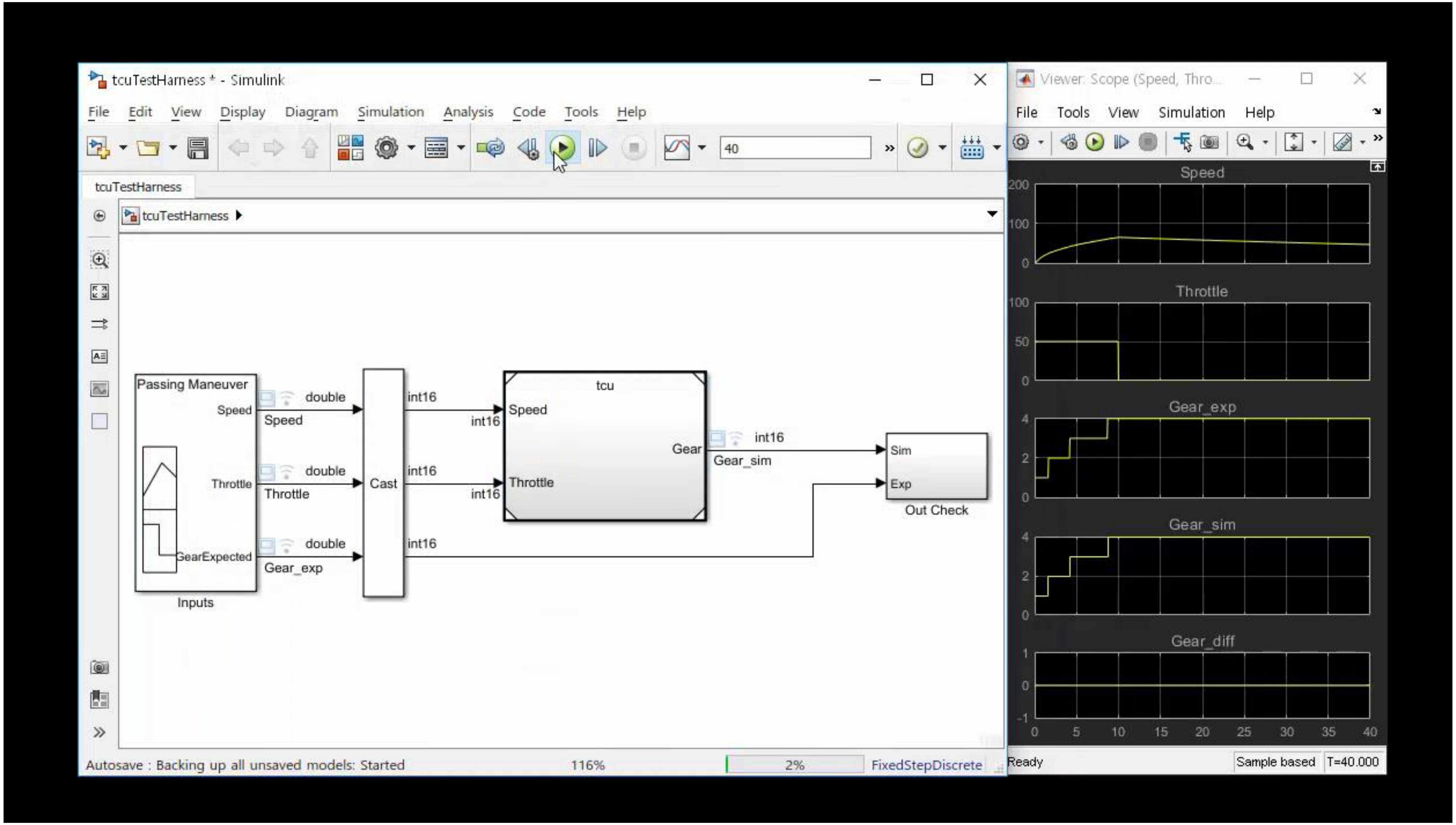


Software-In-the-Loop (SIL) Testing

- Show equivalence, model to code
- Assess code execution time
- Collect code coverage

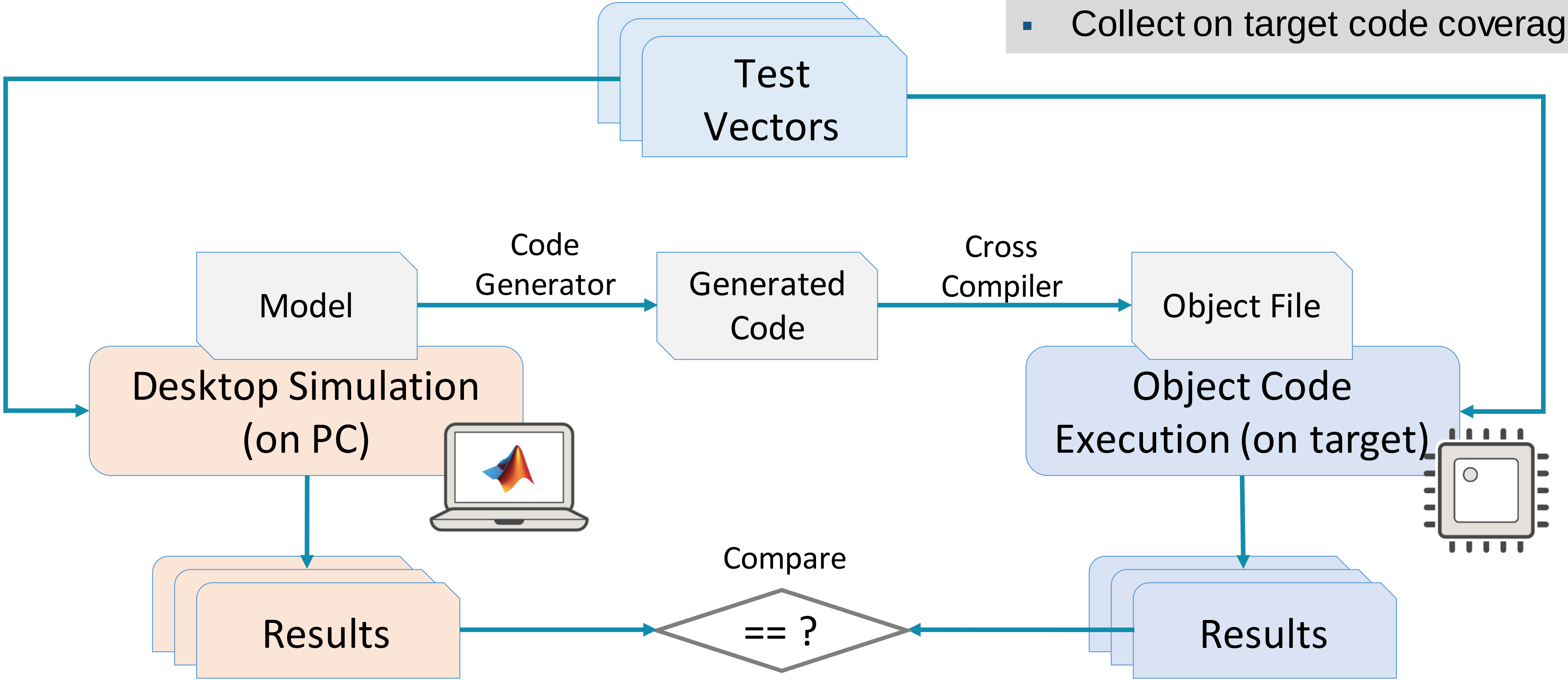


Software-In-the-Loop Test with Model Reference



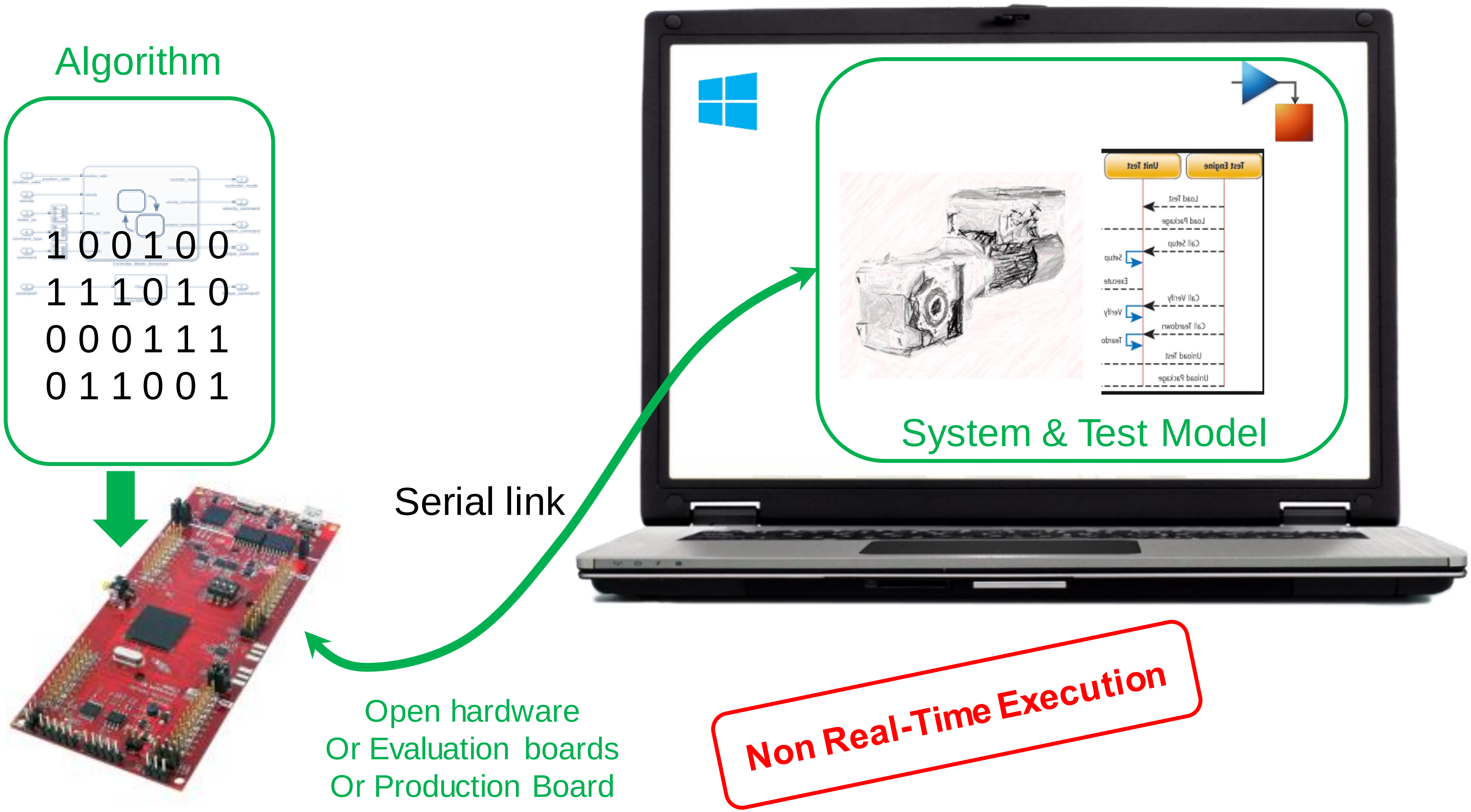
Processor-In-the-Loop (PIL) Testing

- Verify numerical equivalence
- Assess target execution time
- Collect on target code coverage



Processor-In-the-Loop (PIL) Testing

Verify Production Controller with Processor-in-the-loop



Verify and Profile Code Using Processor-In-the-Loop(PIL) Testing

Code Execution Profiling Report for mcb_pmsm_foc_sim_v2/Current Control1

The code execution profiling repor… recorded by instrumentation probe… Profiling for more information.

1. Summary

Total time	
Unit of time	
Command	
Timer frequency (ticks per second…	
Profiling data created	

2. Profiled Sections of Code

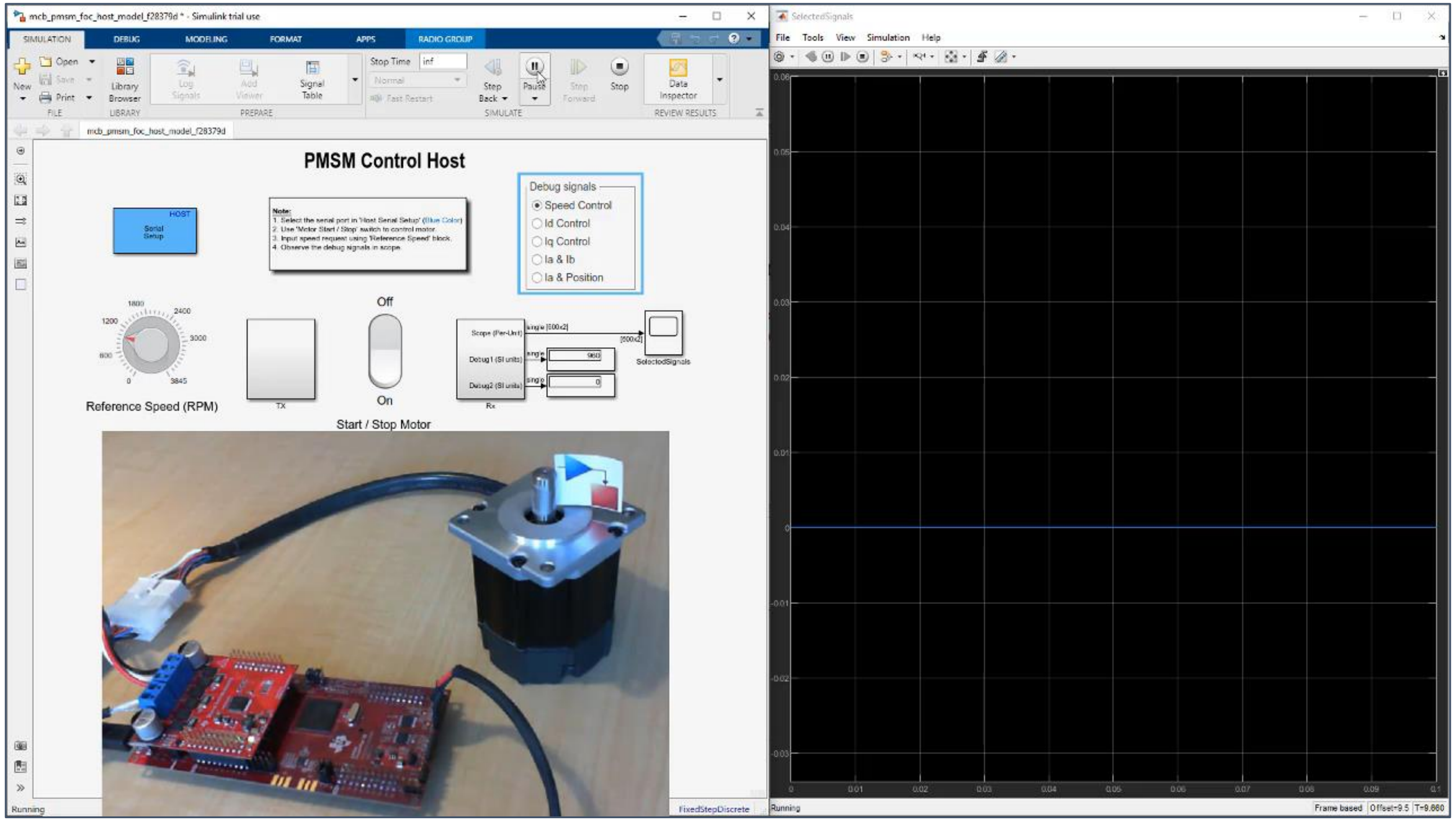
Section	Maximum Execution Time in ns	Average Execution Time in ns	Ma…
[+] Current_initialize	2260	2260	
Current_step [5e-05 0]	5135	5067	
Current_terminate	540	540	

3. CPU Utilization

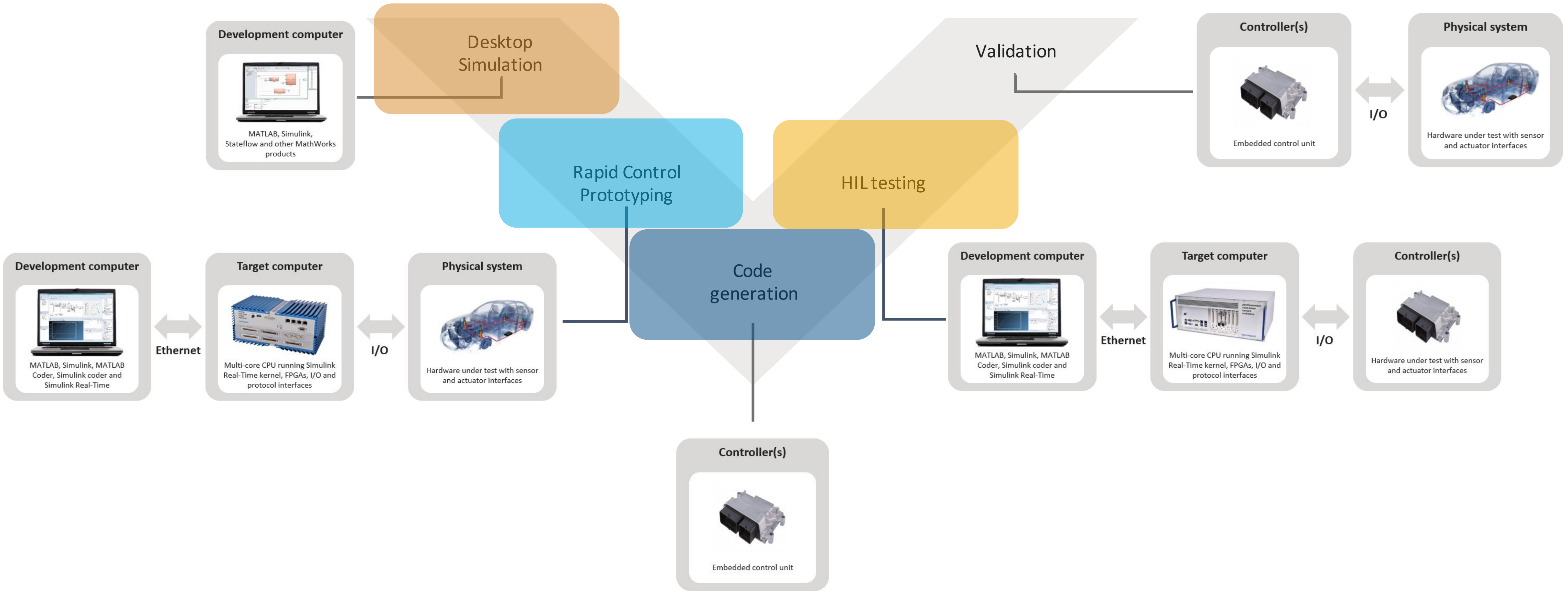
Task	Average CPU Utilization	Maximum CPU Utilization
Current_step [5e-05 0]	10.13%	10.27%
Overall CPU Utilization	10.13%	10.27%

Deployment on the Target

- Generate code (floating and fixed-point)
- Use host model to control and debug
- Validate on hardware



Code Generation and Real-Time Testing in Model-Based Design



Key Takeaways

- Model-based design for motor control enables you to make 50% faster time to market.
  - Various fidelity modeling of motor and inverter using Simscape Electrical
  - Autotuning PI controller gains using optimization algorithm
- Motor Control Toolbox, a new product in R2020a, enables you to minimize development time using reference examples
  - Sensor calibration, built-in algorithmic blocks, automated parameter estimation, and gain-tuning
- Generate, deploy and validate production code

Q&A

MathWorks®