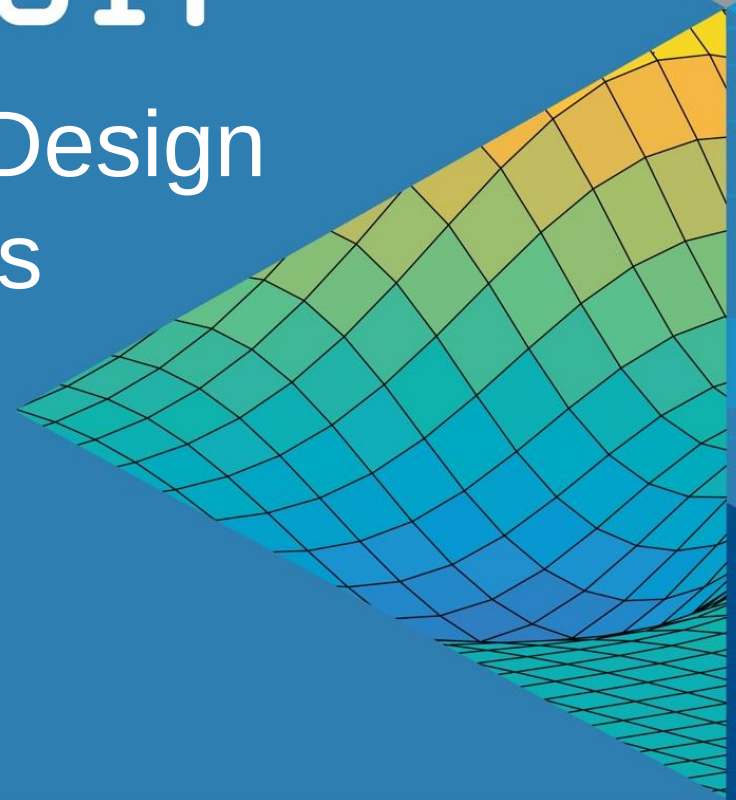
MATLAB EXPO 2017

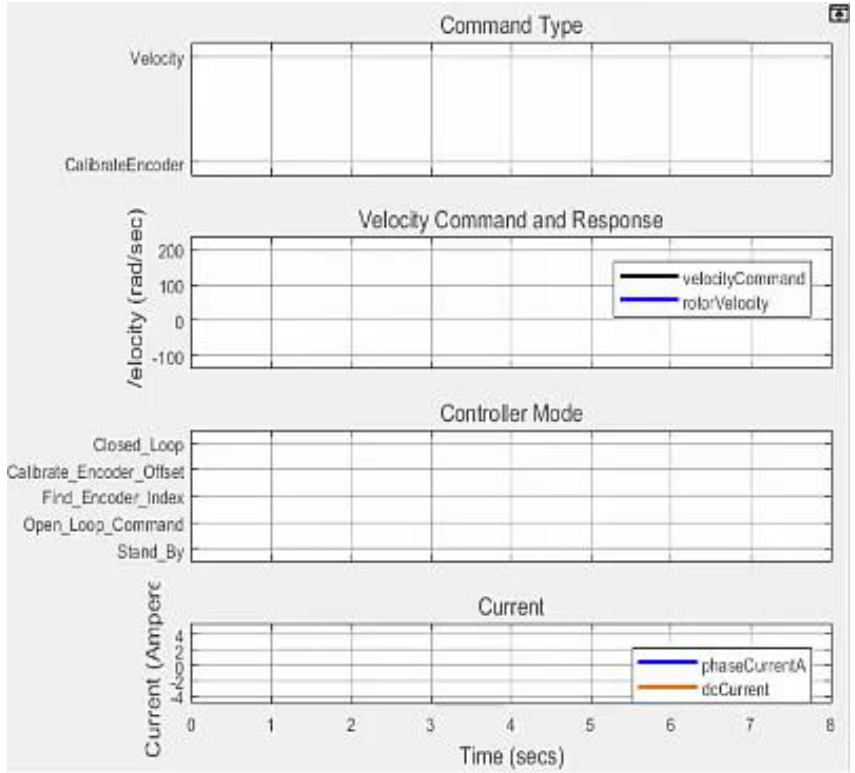
Hardware and Software Co-Design for Motor Control Applications

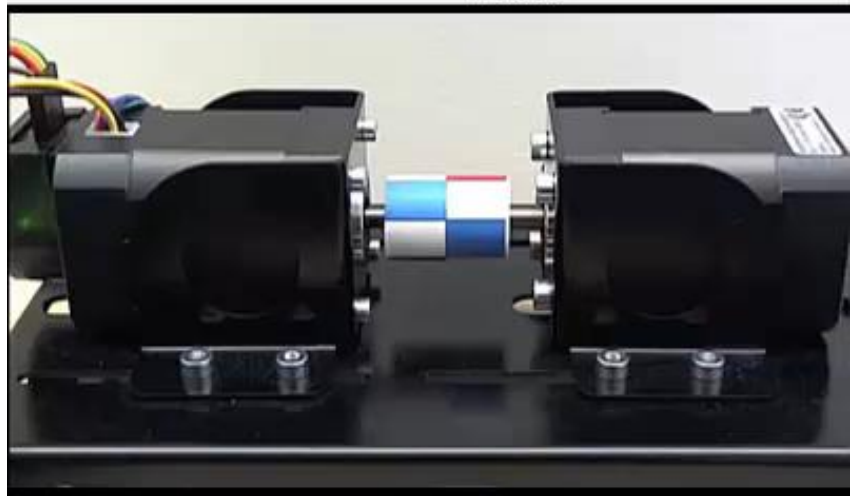


GianCarlo Pacitti
Senior Application Engineer, MathWorks

Agenda

- Why use Hardware and Software for motor control?
- Why use Model-Based Design for motor control?
- How to use Model-Based Design for motor control?



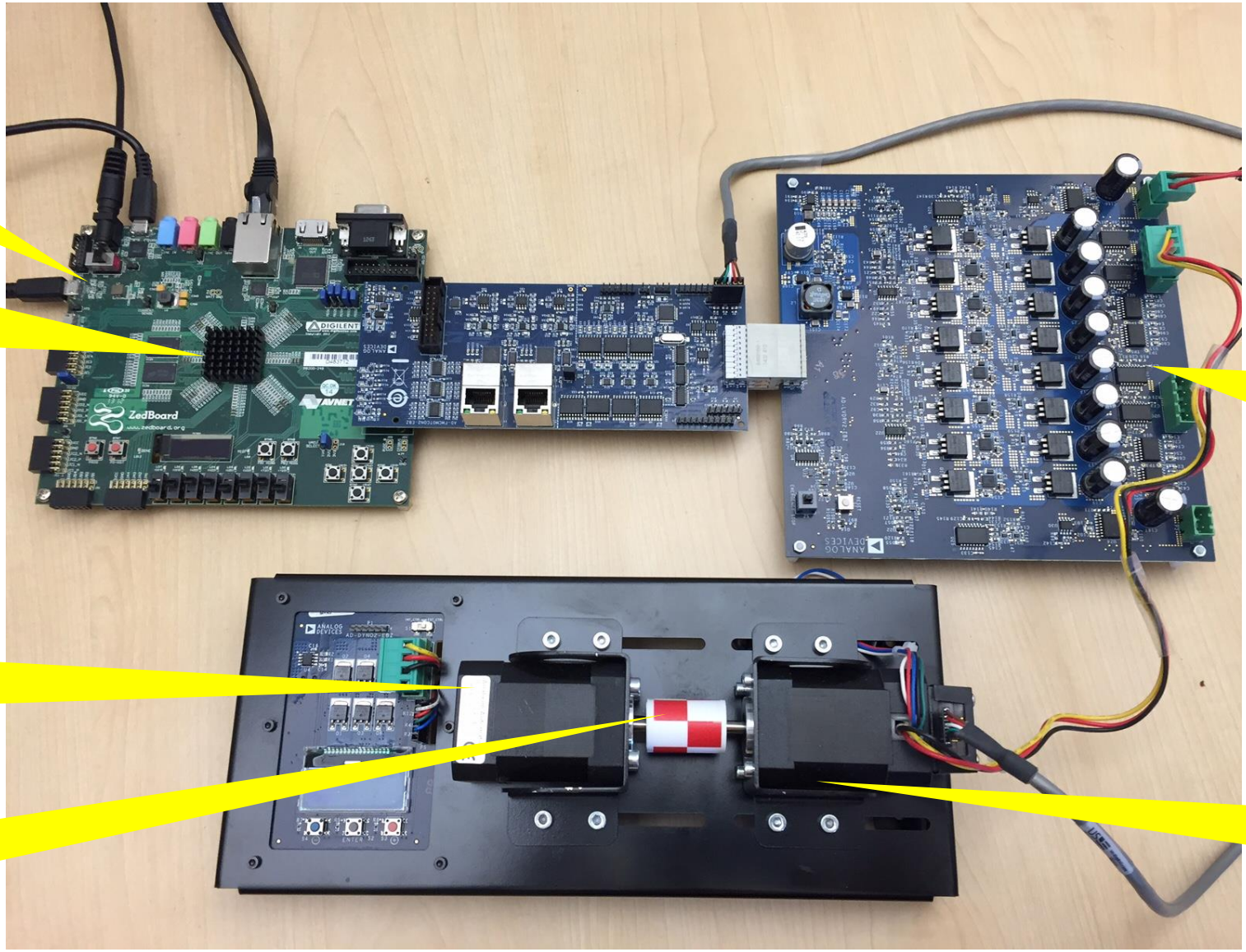


ZedBoard

Zynq SoC
(XC7Z020)

Load motor

Mechanical coupler



FMC module: control board + low-voltage board

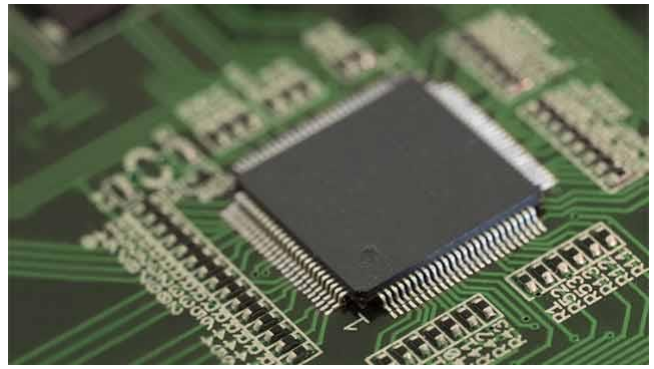
Motor under test (with encoder)

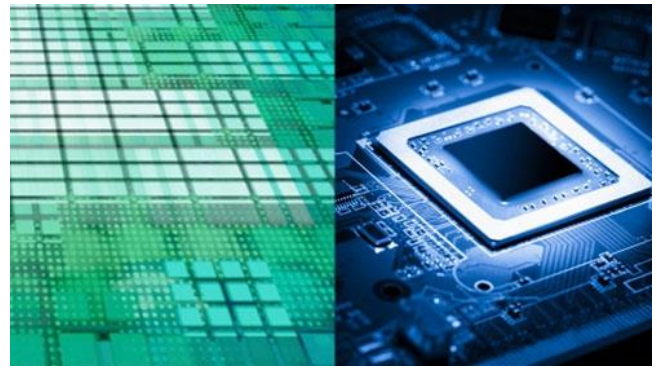
Key trend: Increasing demands from motor drives

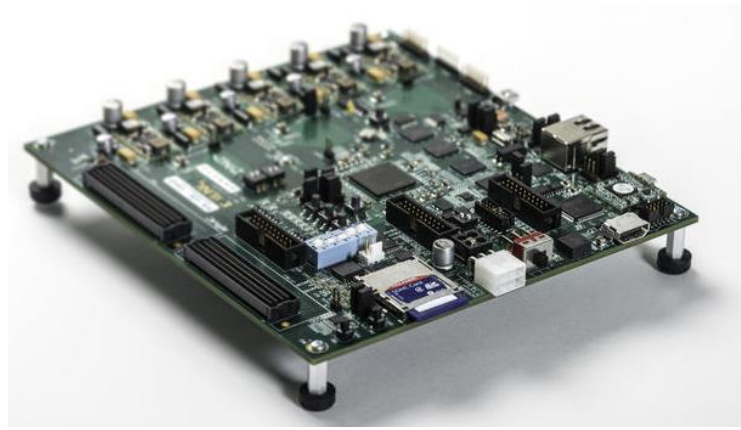
- Increased performance targets require advanced algorithms
- Advanced algorithms require faster computing performance.
  - Field-Oriented Control
  - Sensorless motor control
  - Vibration detection and suppression
  - Multi-axis control

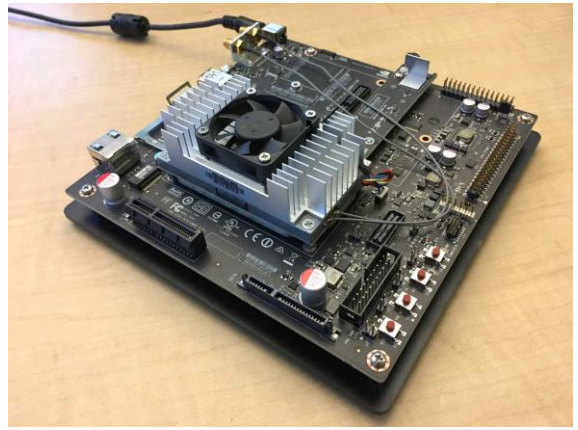
Where are algorithms being run to gain performance?

- Multi-core microprocessors
- Multi-processor systems
- FPGAs
- ASICs
- System-on-Chip devices (SoCs)
- GPUs*







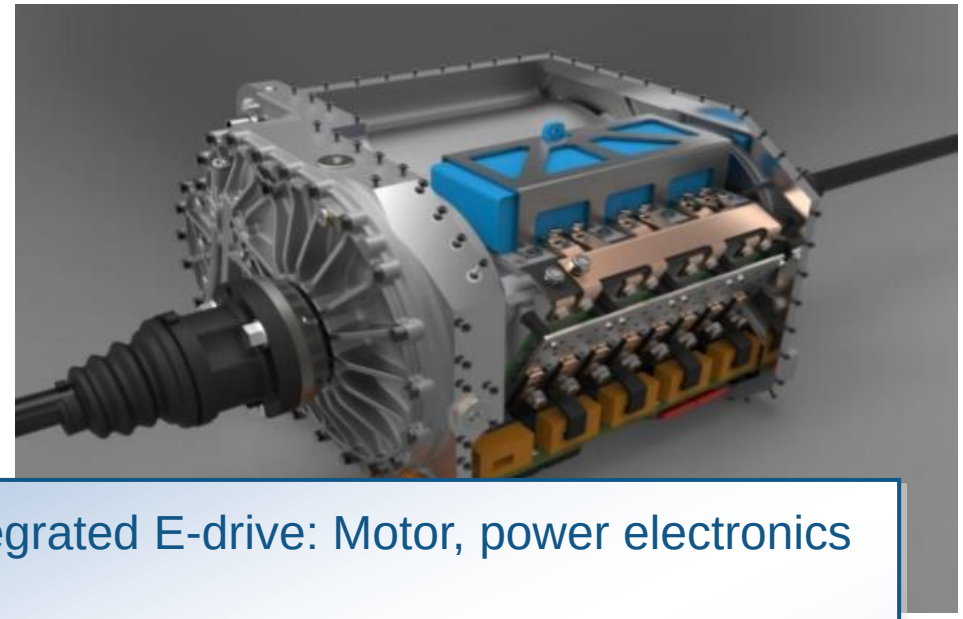


- **<u>Hardware and Software algorithms must be designed together</u>**

*particularly for vision applications

Punch Powertrain develops complex SoC-based motor control

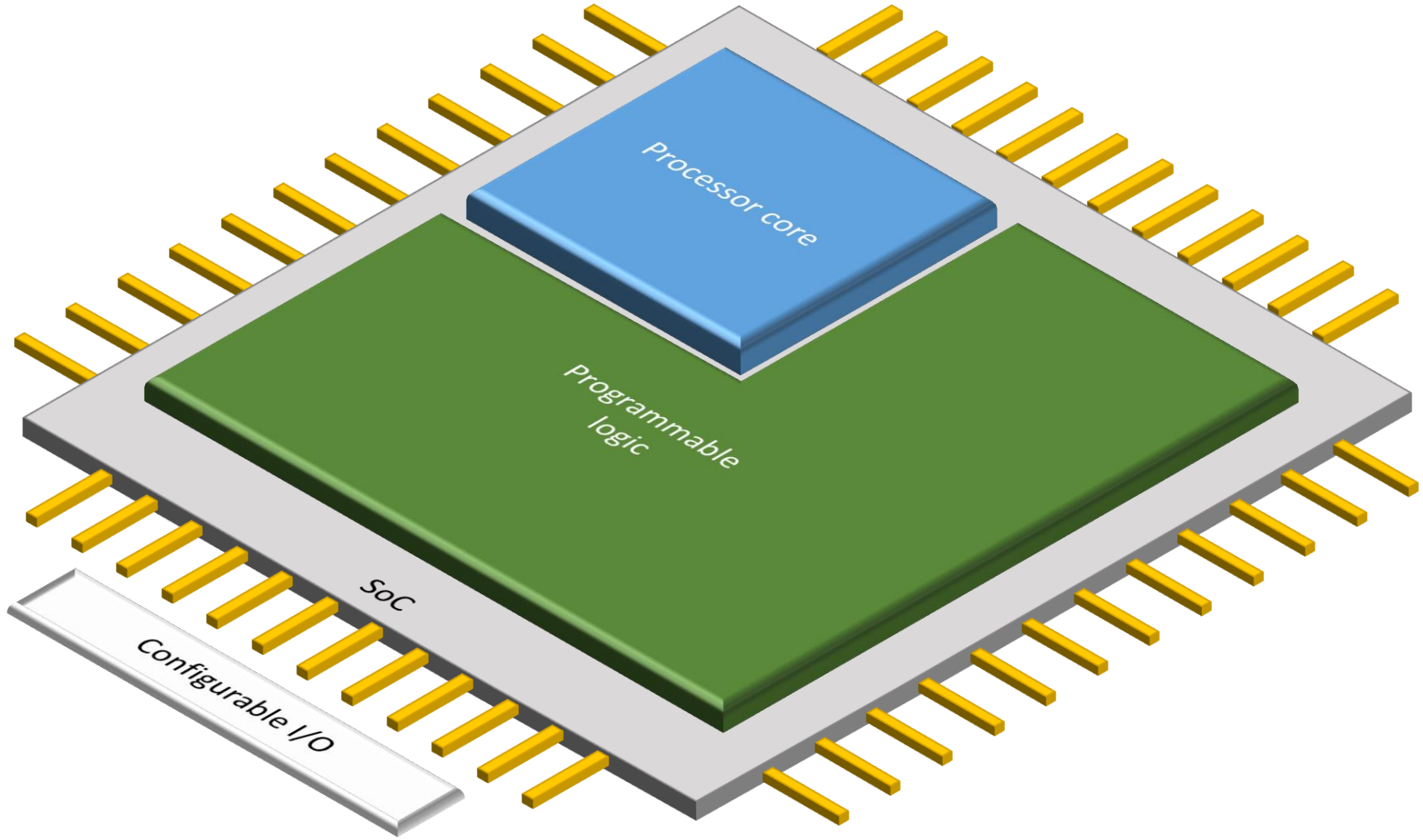
- Powertrains for hybrid and electric vehicles
- Need to increase power density and efficiency at a reduced cost
  - Integrate motor and power electronics in the transmission
- New switched reluctance motor
  - Fast: 2x the speed of their previous motor
    - Target to a Xilinx® Zynq® SoC 7045 device
  - Complex: 4 different control strategies
- Needed to get to market quickly
- No experience designing FPGAs!



- ✓ Designed integrated E-drive: Motor, power electronics and software
- ✓ 4 different control strategies implemented
- ✓ Done in 1.5 years with 2FTE's
- ✓ Models reusable for production
- ✓ Smooth integration and validation due to development process – thorough validation before electronics are produced and put in the testbench

Link to video

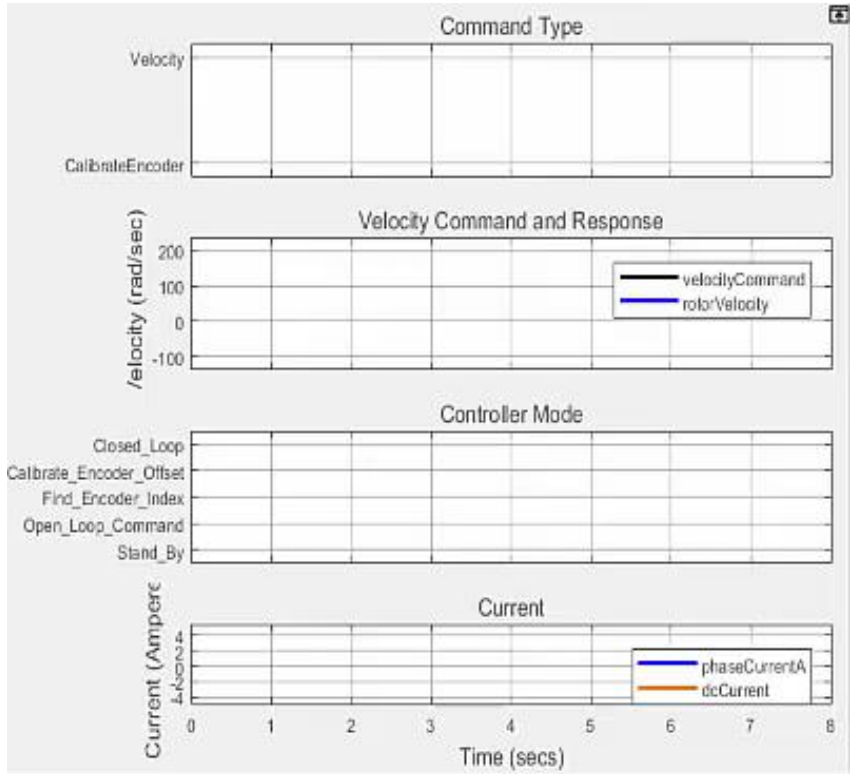
What's Inside an FPGA SoC?

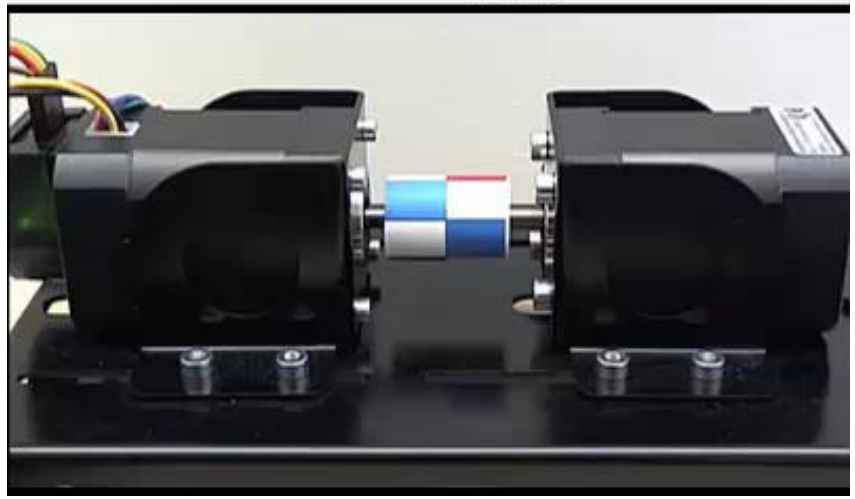


Why use Hardware and Software for Motor Control?

- In order to meet increased performance
- You need more complex algorithms
- Running on the right hardware

- Why use Hardware and Software for motor control?
- Why use Model-Based Design for motor control?
- How to use Model-Based Design for motor control?





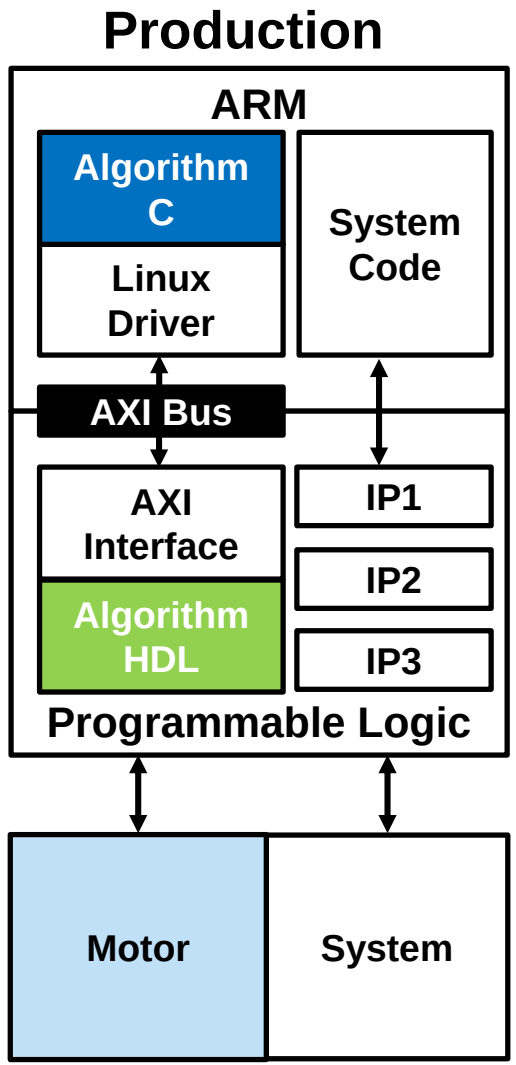
Challenges in Developing Advanced Motor Control Algorithms

- Integration requires collaboration
- Validation of design specifications with limits on access to test hardware
- How to make design decisions?

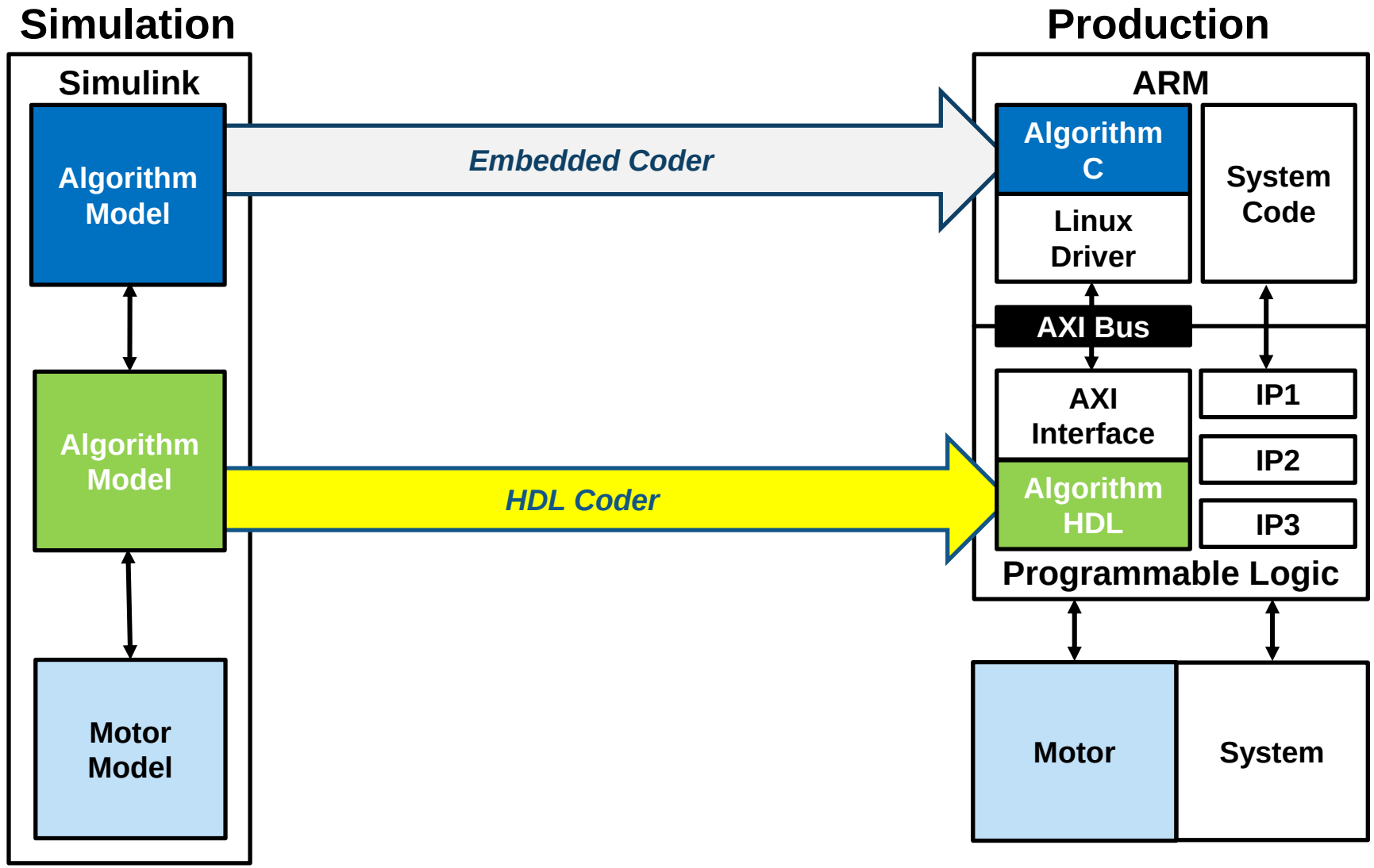
Why use Model-Based Design to Develop Motor Control Applications?

- Enables early validation of specifications using simulation months before hardware is available.
- Dramatically improves design team collaboration and designer productivity by using a single design environment.
- Reduces hardware testing time by 5x by shifting design from lab to the desktop

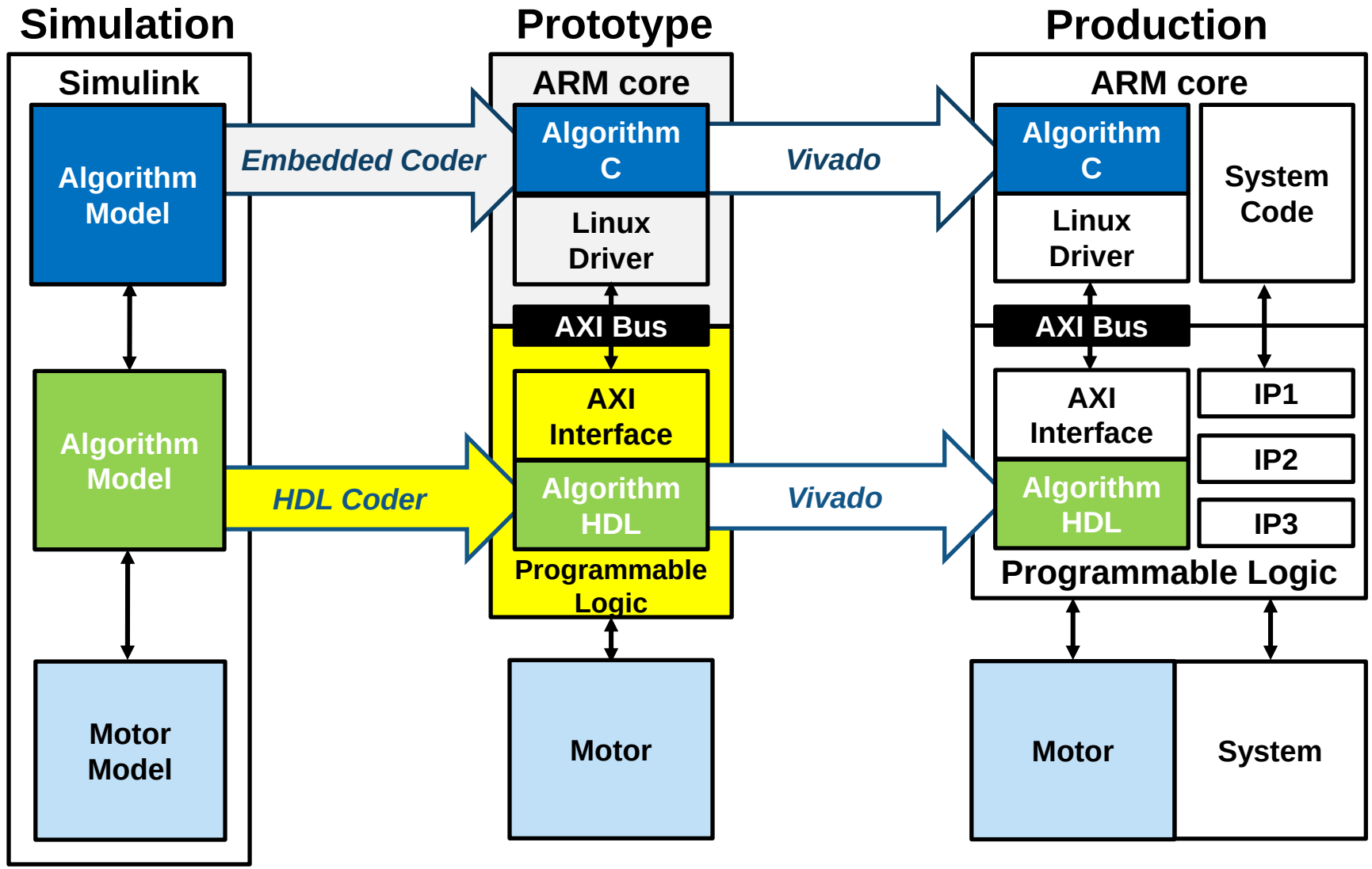
Components of Motor Control Production Applications



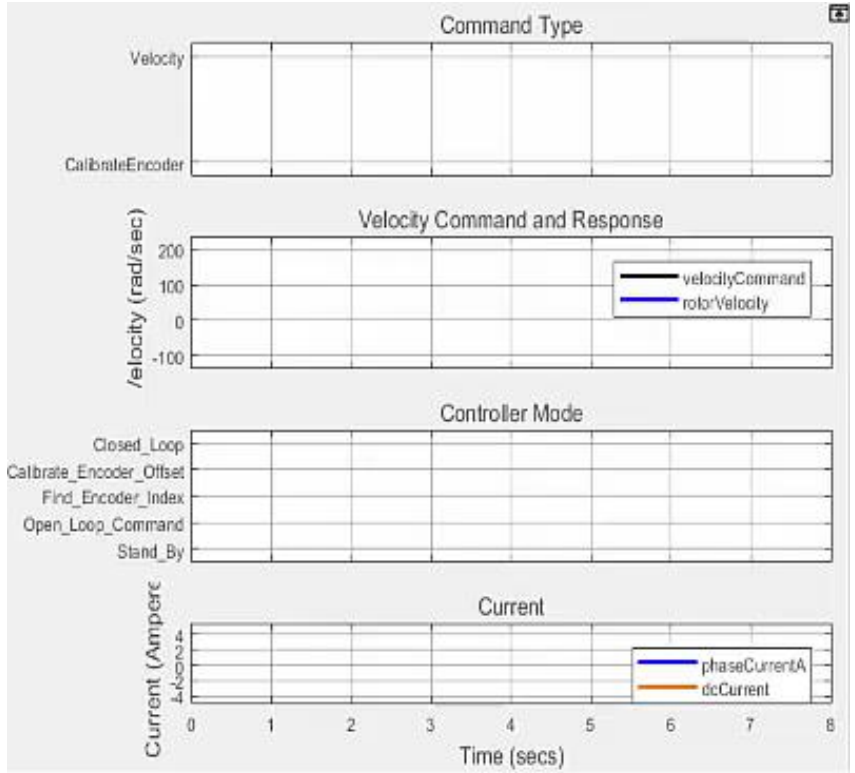
From Simulation to Production

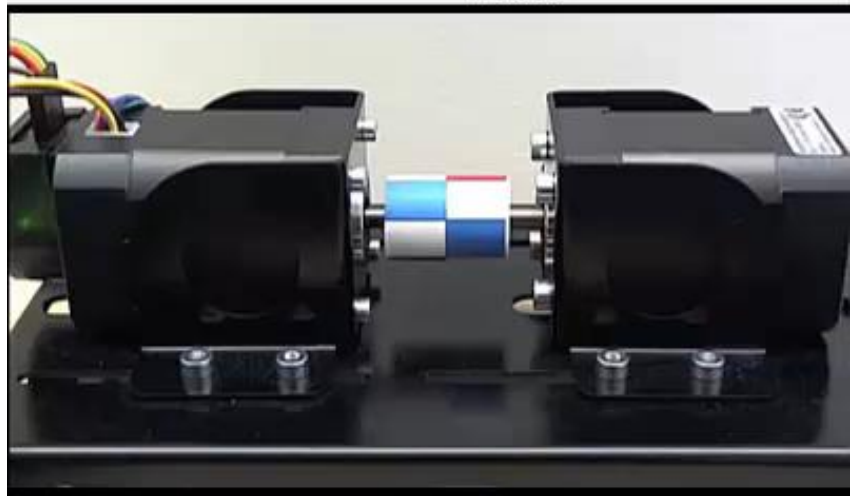


From Simulation to Prototype to Production



- Why use Hardware and Software for motor control?
- Why use Model-Based Design for motor control?
- How to use Model-Based Design for motor control?



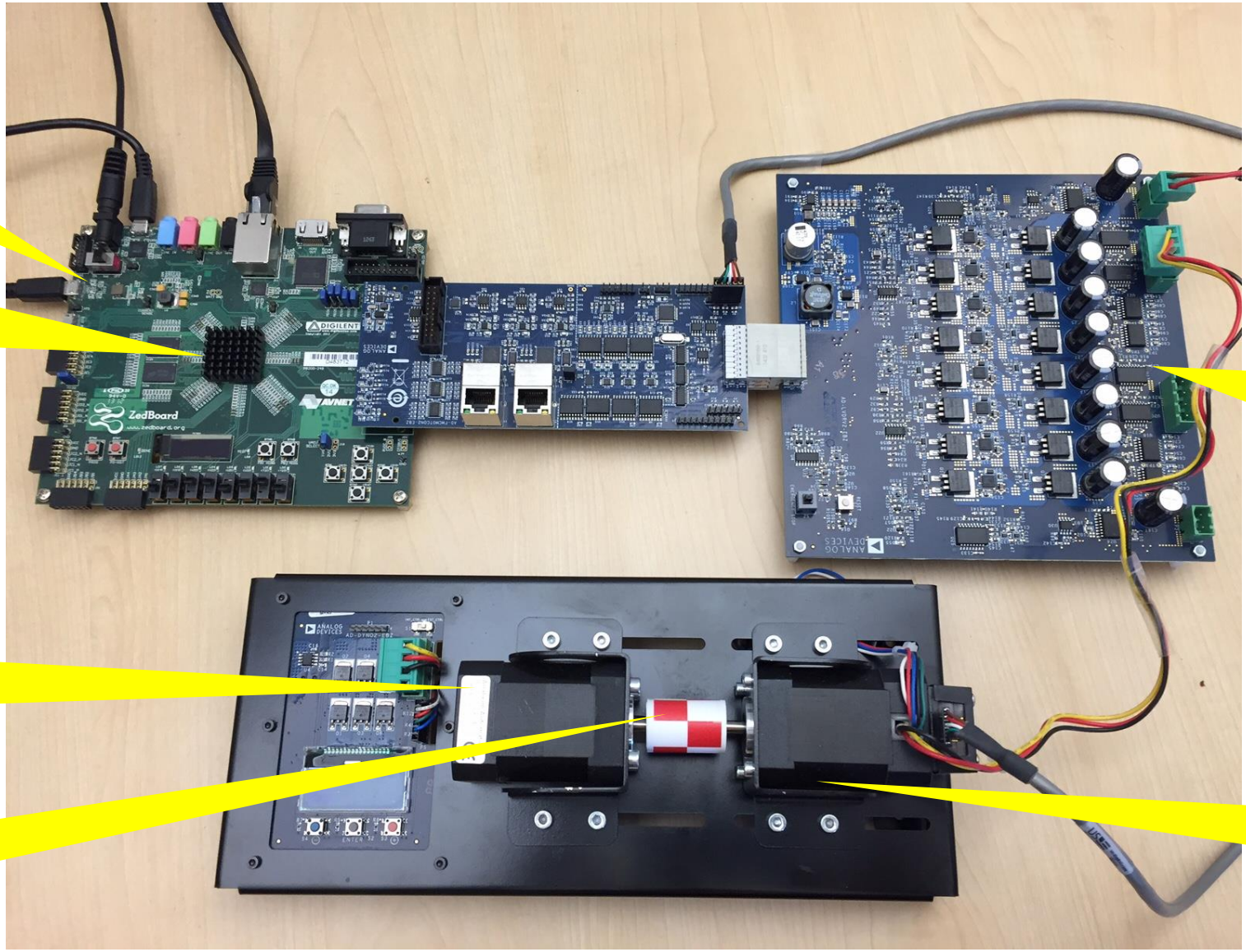


ZedBoard

Zynq SoC
(XC7Z020)

Load motor

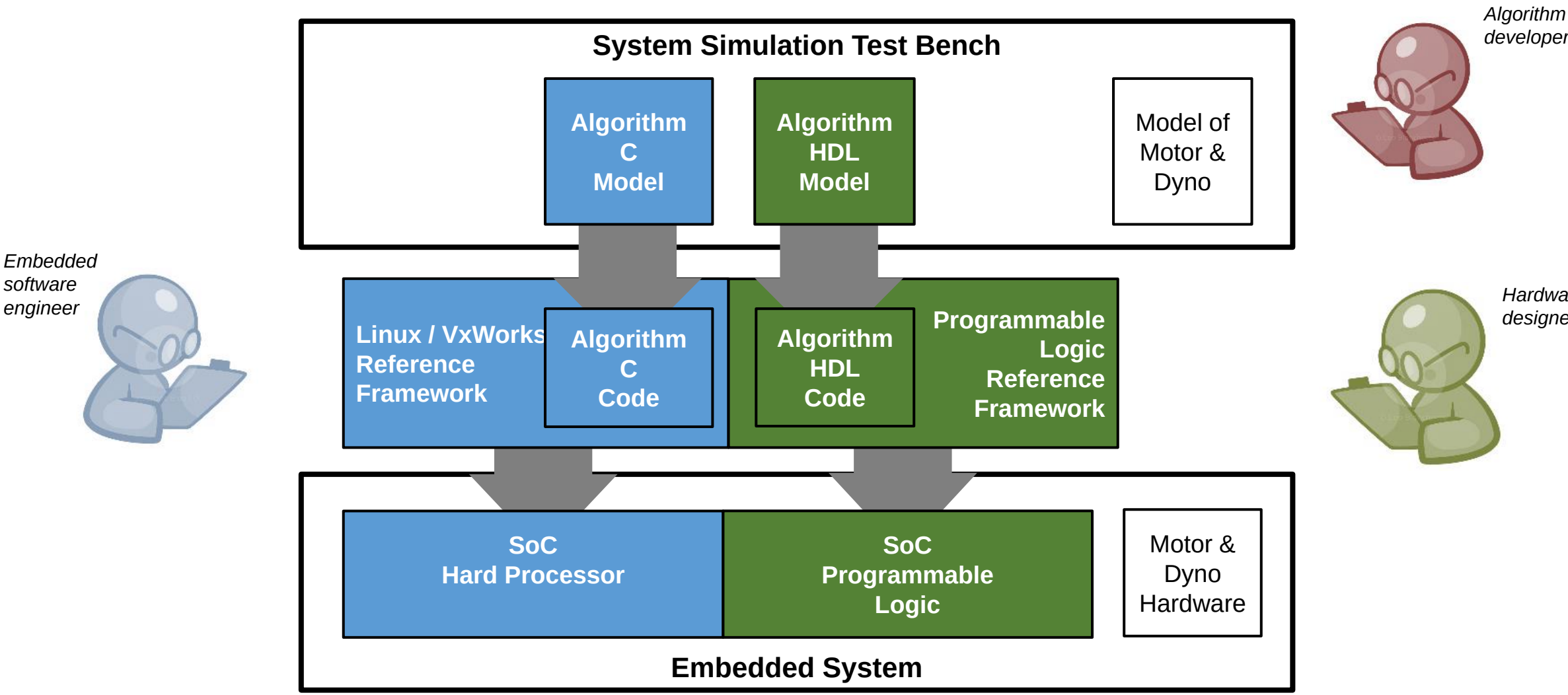
Mechanical coupler

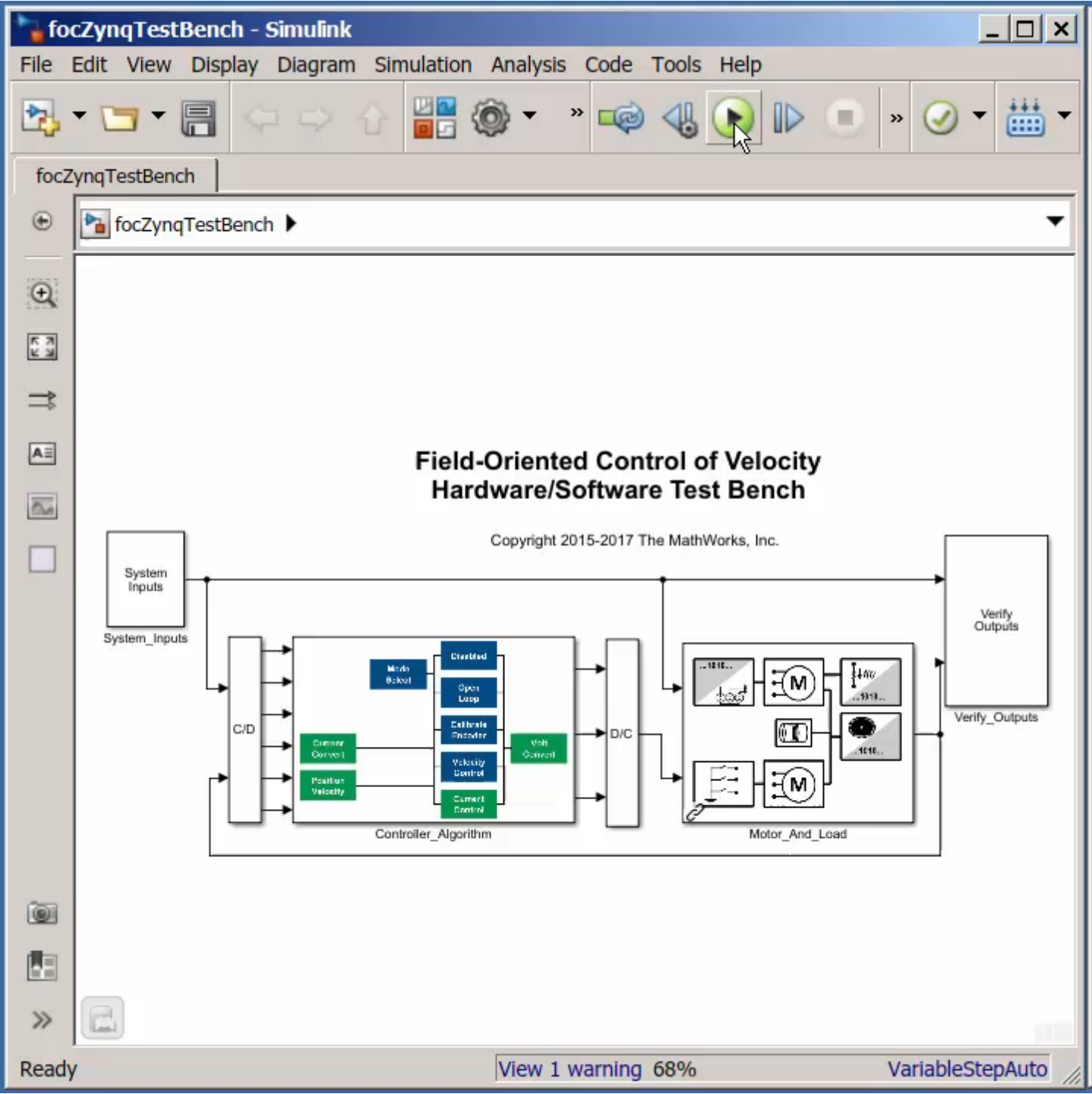


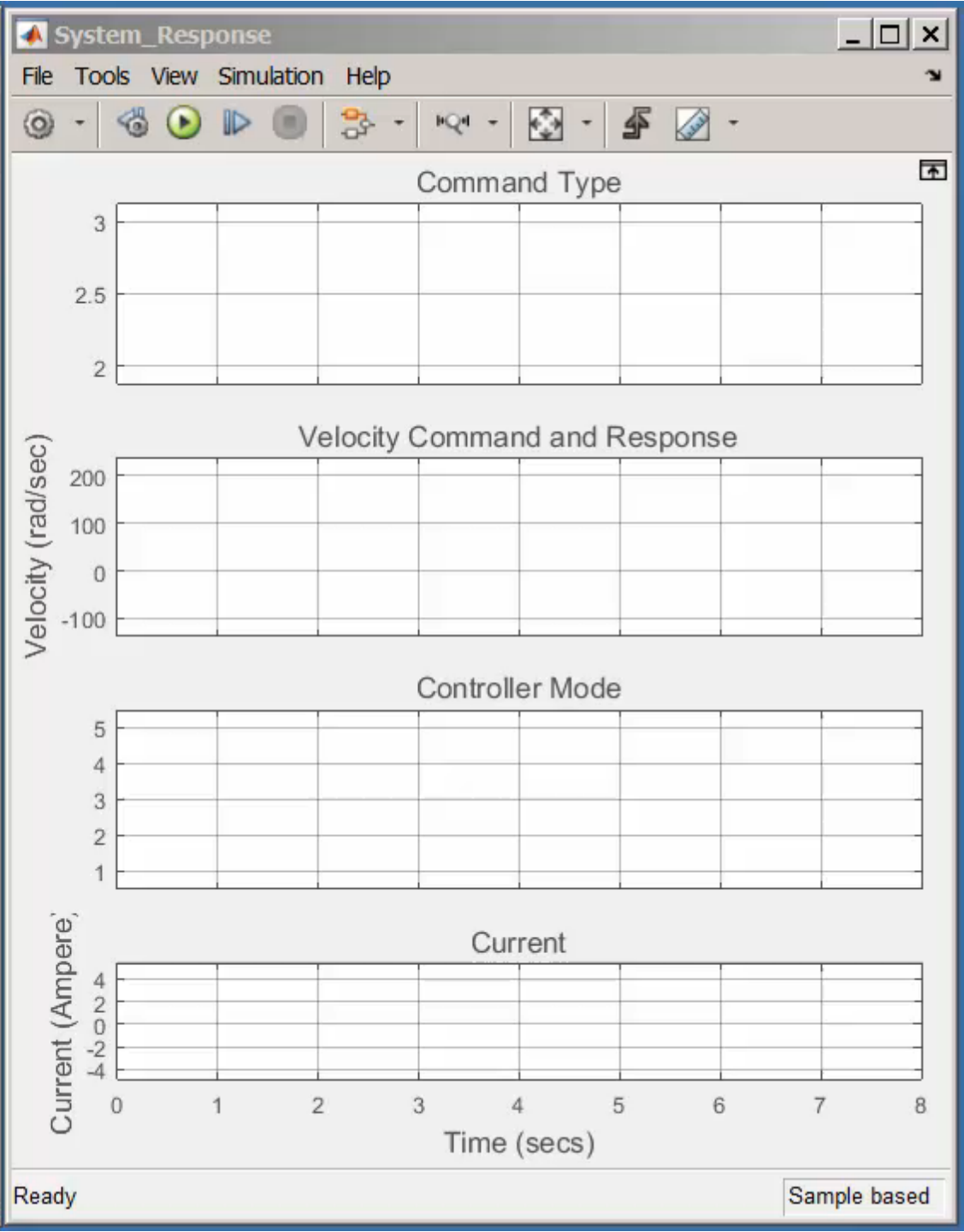
FMC module: control board + low-voltage board

Motor under test (with encoder)

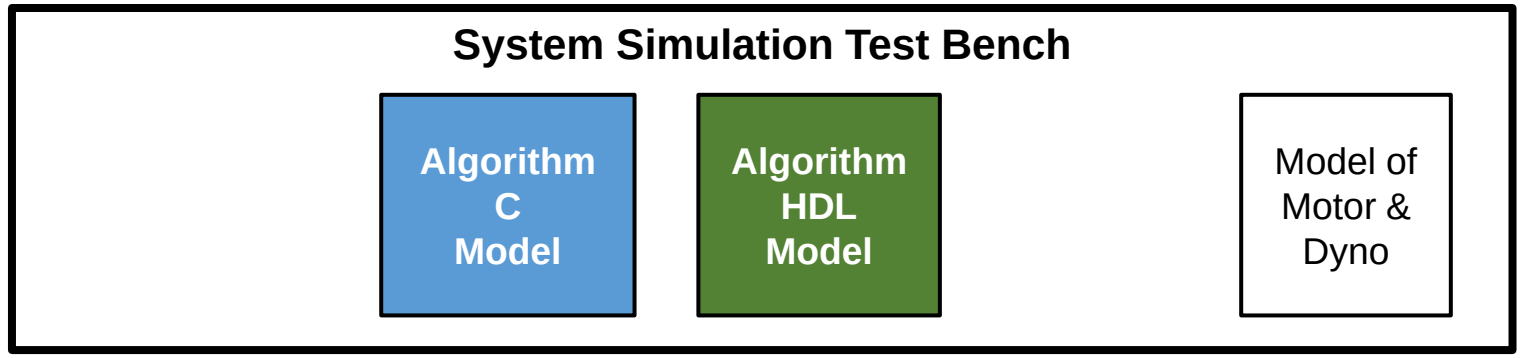
Conceptual workflow targeting hardware and software

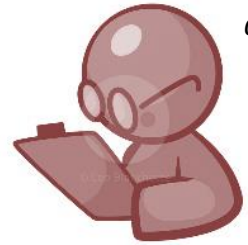



focZynqTestBench - Simulink
File Edit View Display Diagram Simulation Analysis Code Tools Help
focZynqTestBench
focZynqTestBench
Field-Oriented Control of Velocity
Hardware/Software Test Bench
Copyright 2015-2017 The MathWorks, Inc.
System Inputs
System_Inputs
C/D
Controller_Algorithm
D/C
Motor_And_Load
Verify Outputs
Verify_Outputs
Ready
View 1 warning 68%
VariableStepAuto


System_Response
File Tools View Simulation Help
Command Type
Velocity Command and Response
Velocity (rad/sec)
Controller Mode
Current
Current (Ampere)
Time (secs)
Ready
Sample based

Building a System Simulation Test Bench

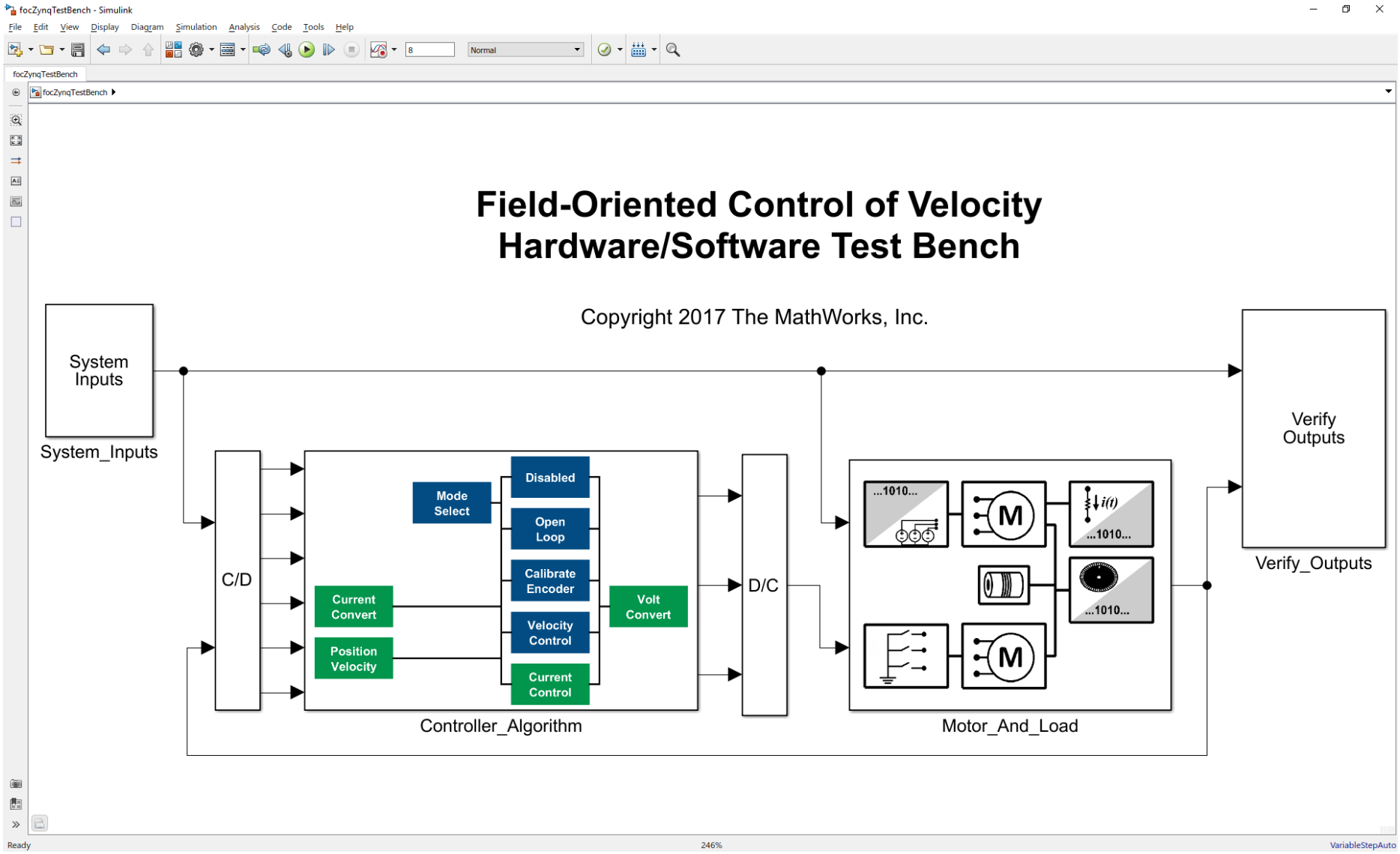




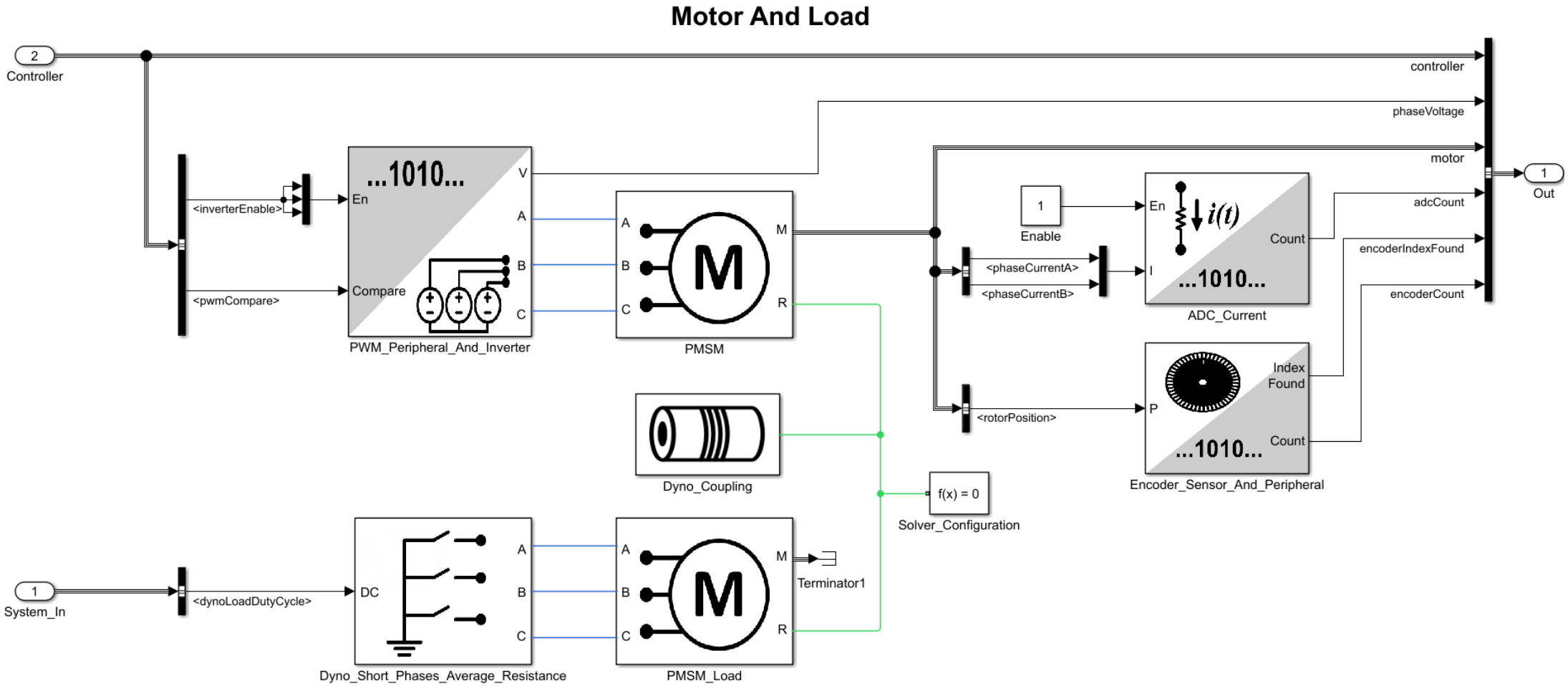
Algorithm developer

- How do I get a good model of the motor?
- How can I make sure it matches real-world behaviour?

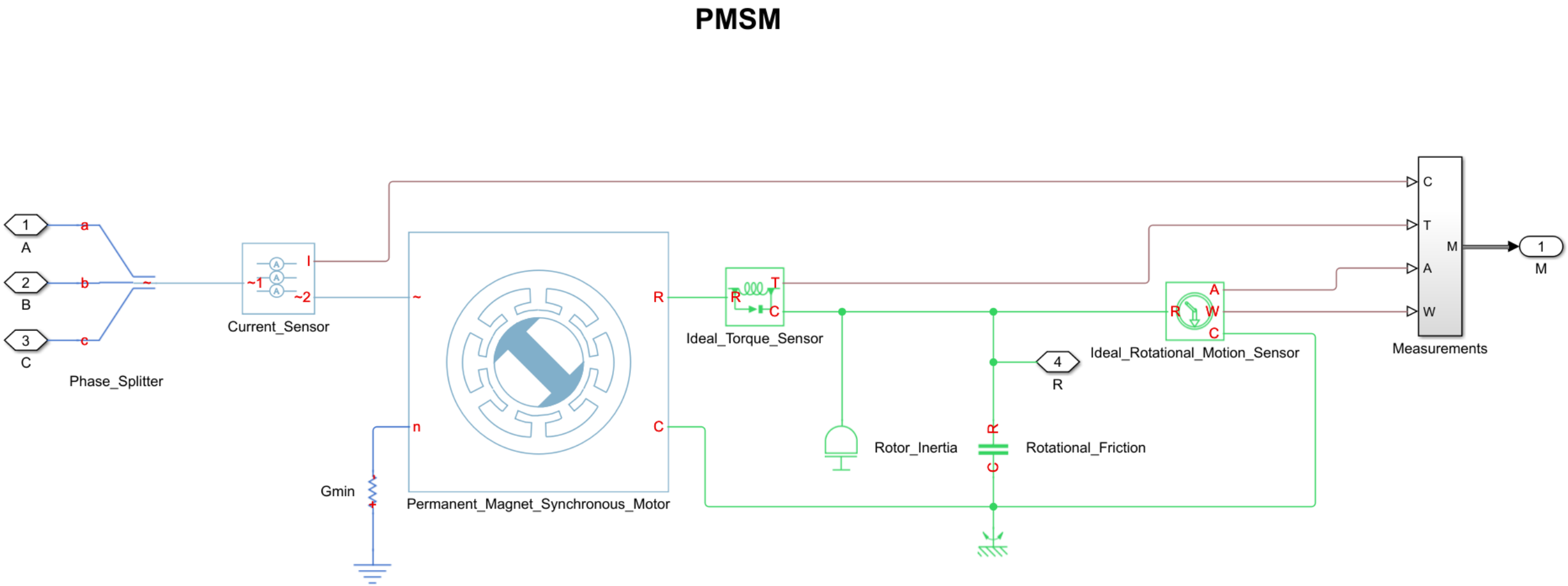
What's Inside a Motor Model?



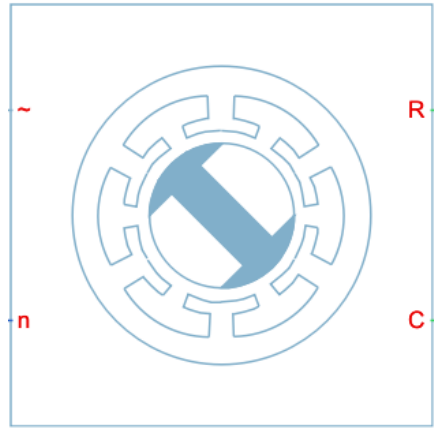
What's Inside a Motor Model?



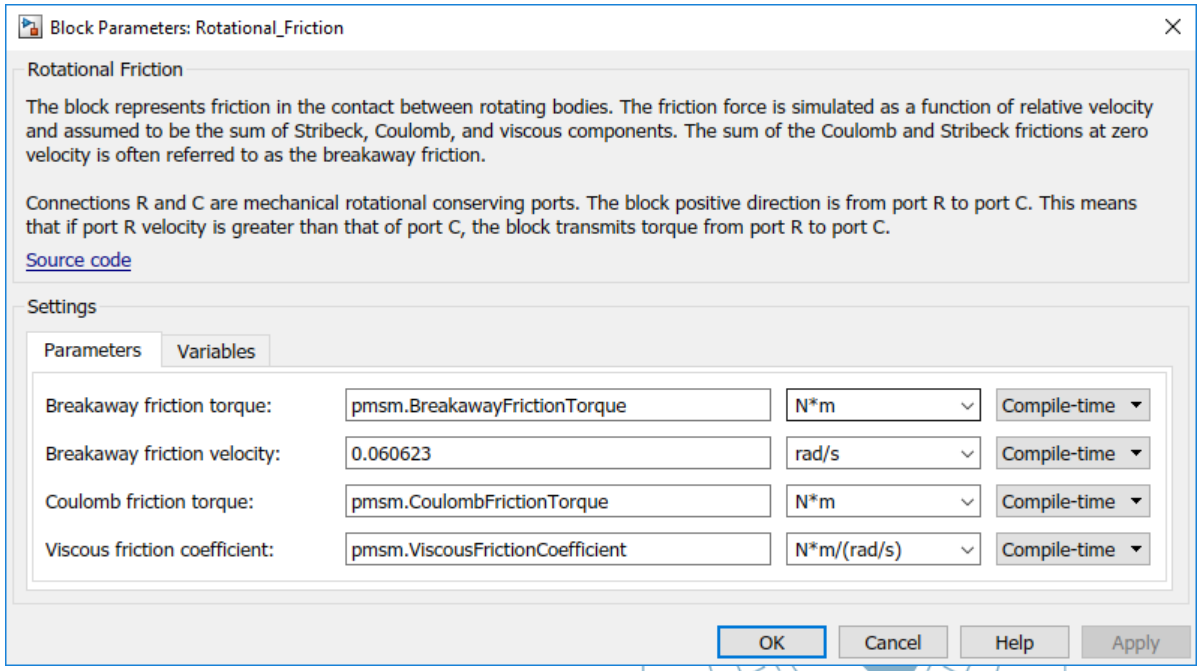
What's Inside a Motor Model?



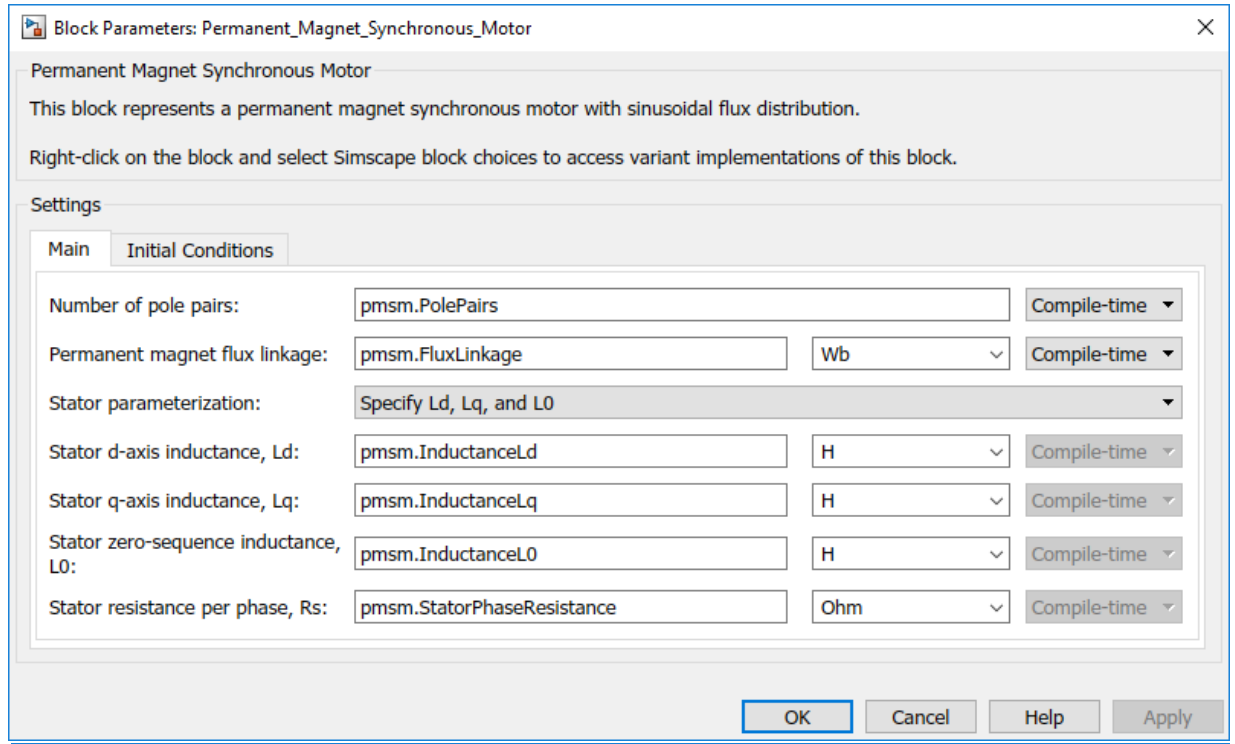
What's Inside a Motor Model?

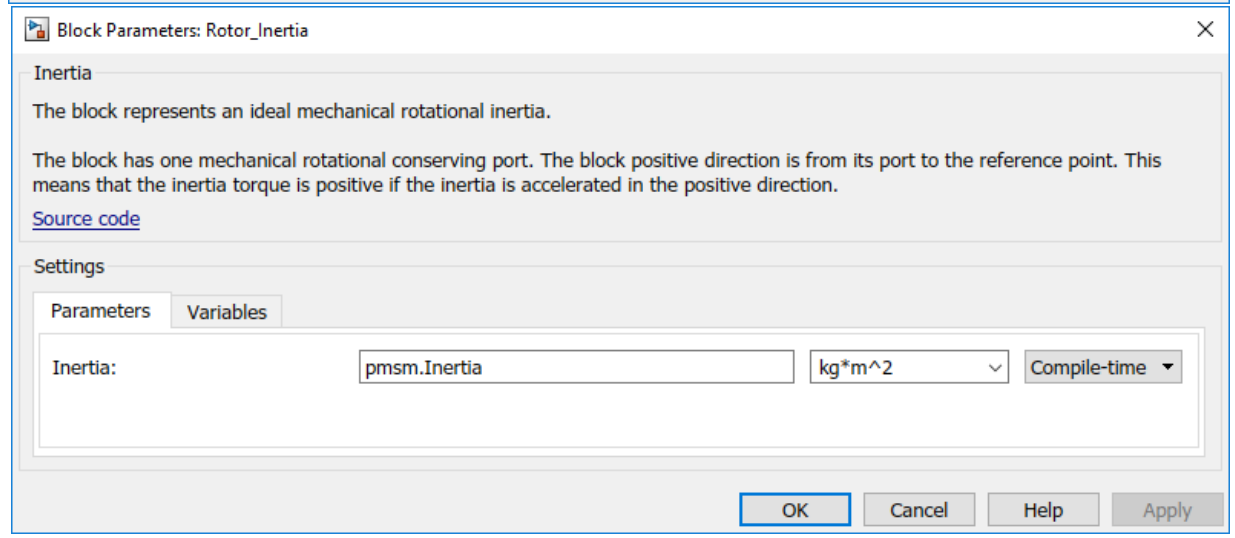


What's Inside a Motor Model?



Permanent_Magnet_Synchronous_Motor





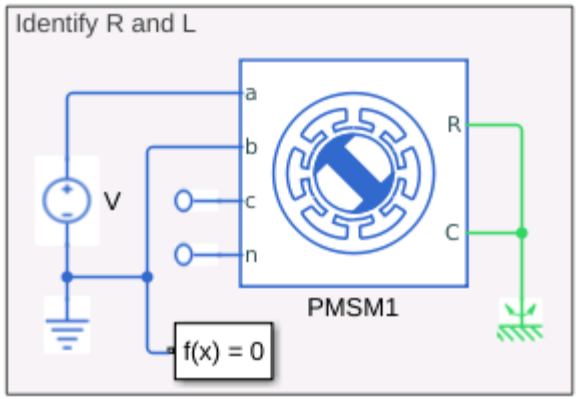
- How can we find the parameters we need for the model?

How to Find the Right Motor Parameters?

- Ask the motor designer
- From manufacturer's data sheets
- From direct bench-top measurements or test data

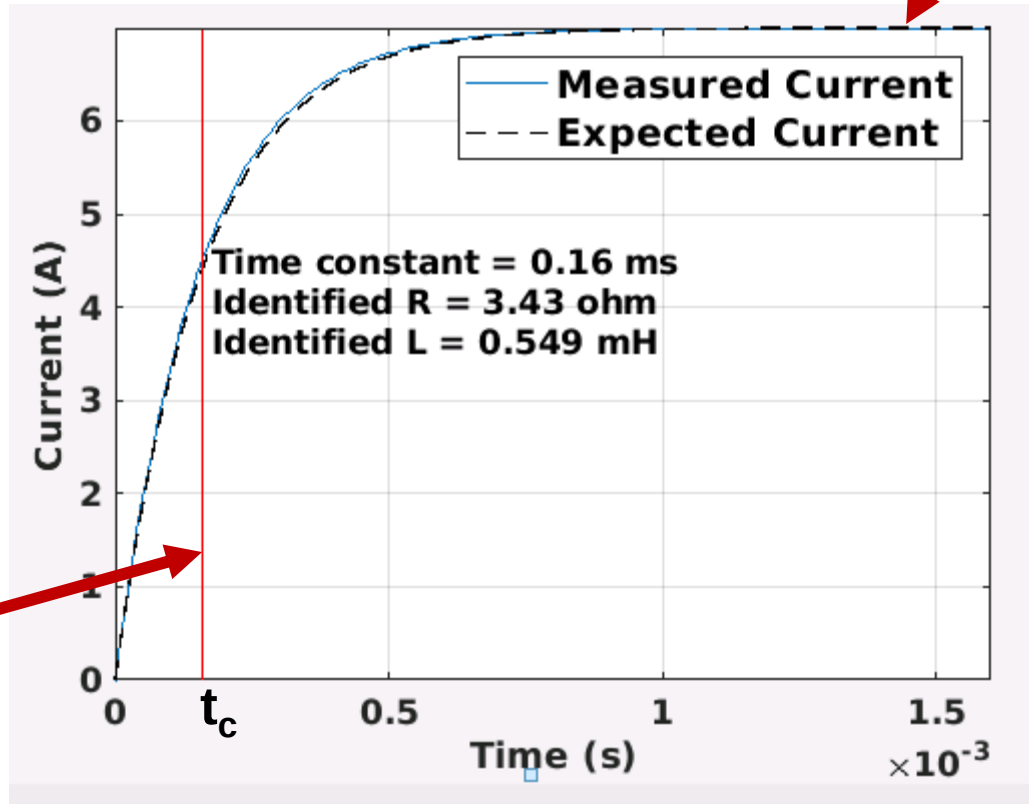
Modelling a PMSM with limited supplier data

Tune to measurement data – Step 1



Locked rotor
Step voltage test

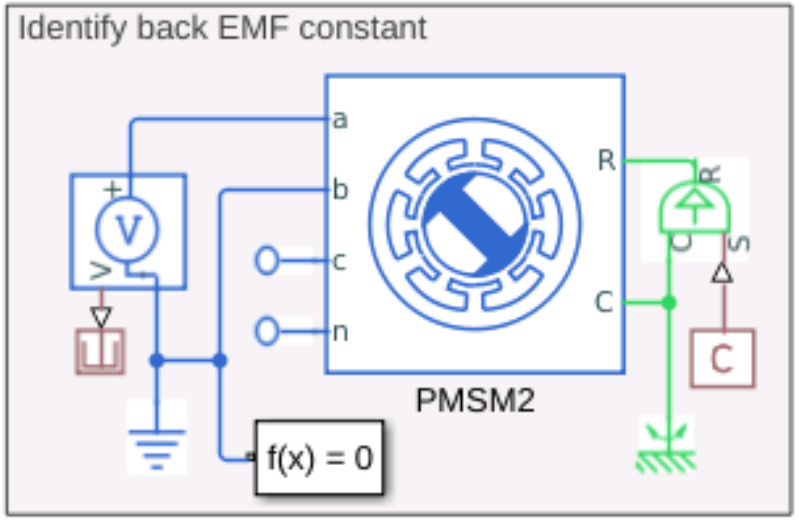
R=V/I



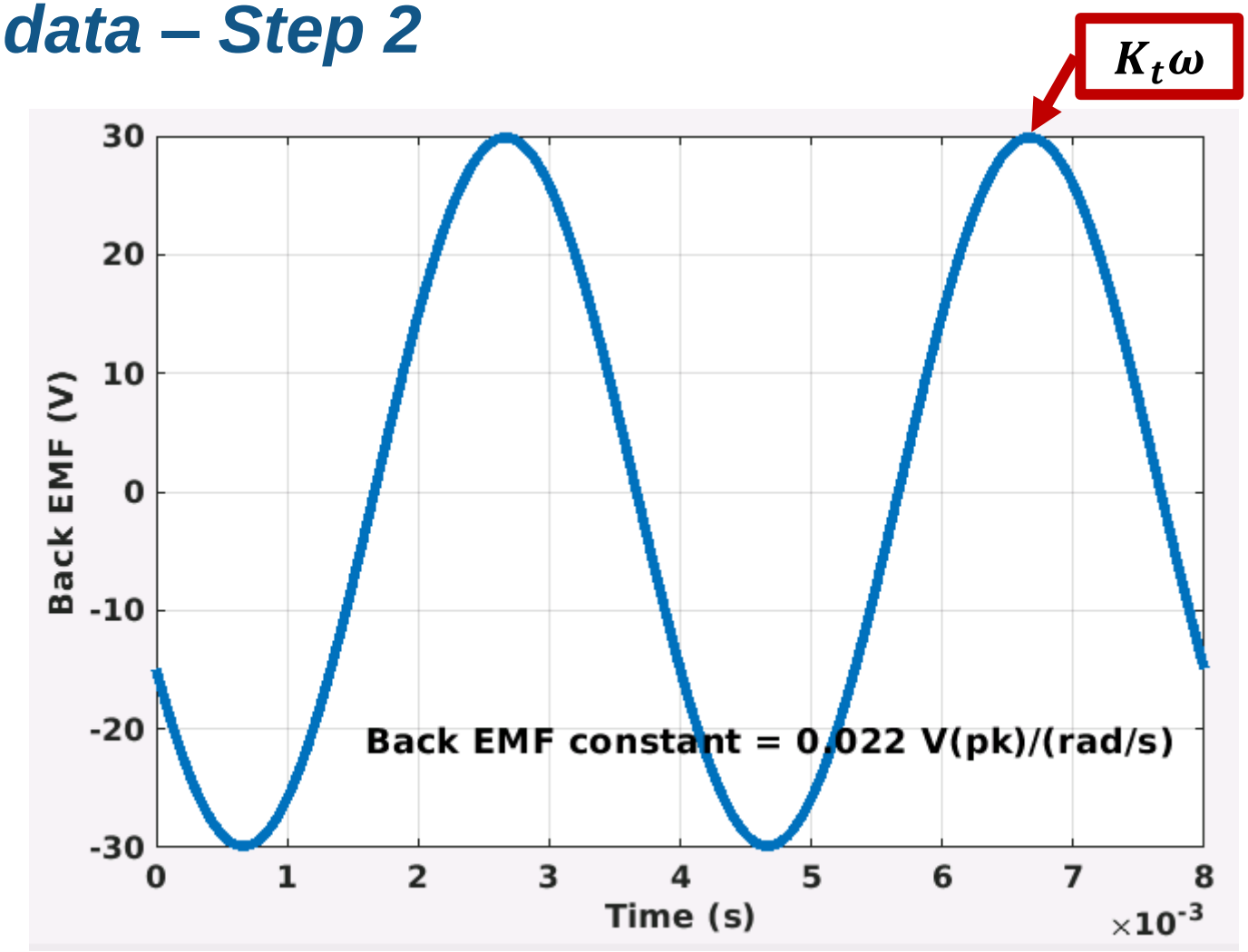
$L=R*t_c$

Modelling a PMSM with limited supplier data

Tune to measurement data – Step 2

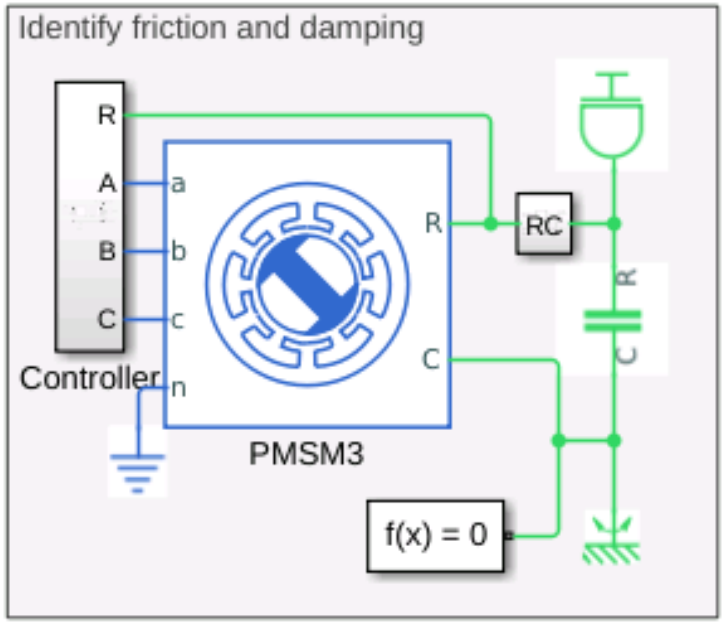


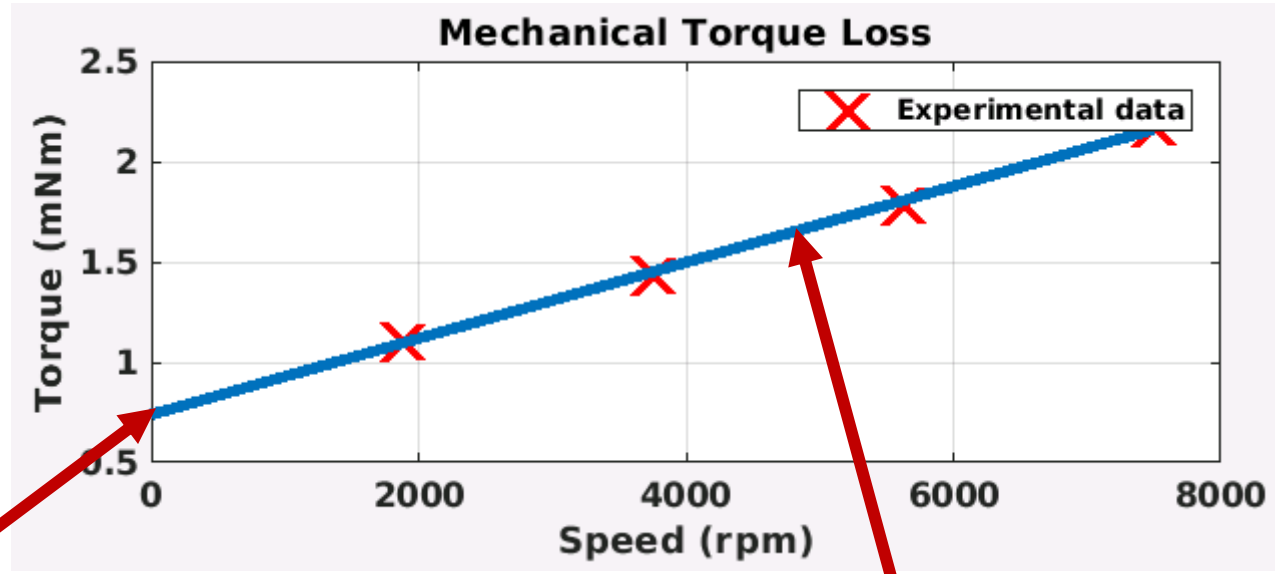
Back EMF test



Modelling a PMSM with limited supplier data

Tune to measurement data – Step 3

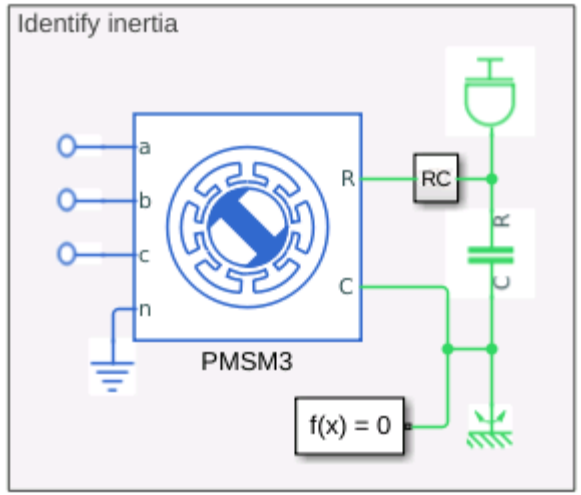




Friction

Damping coefficient = gradient

Modelling a PMSM with limited supplier data
Tune to measurement data – Step 4

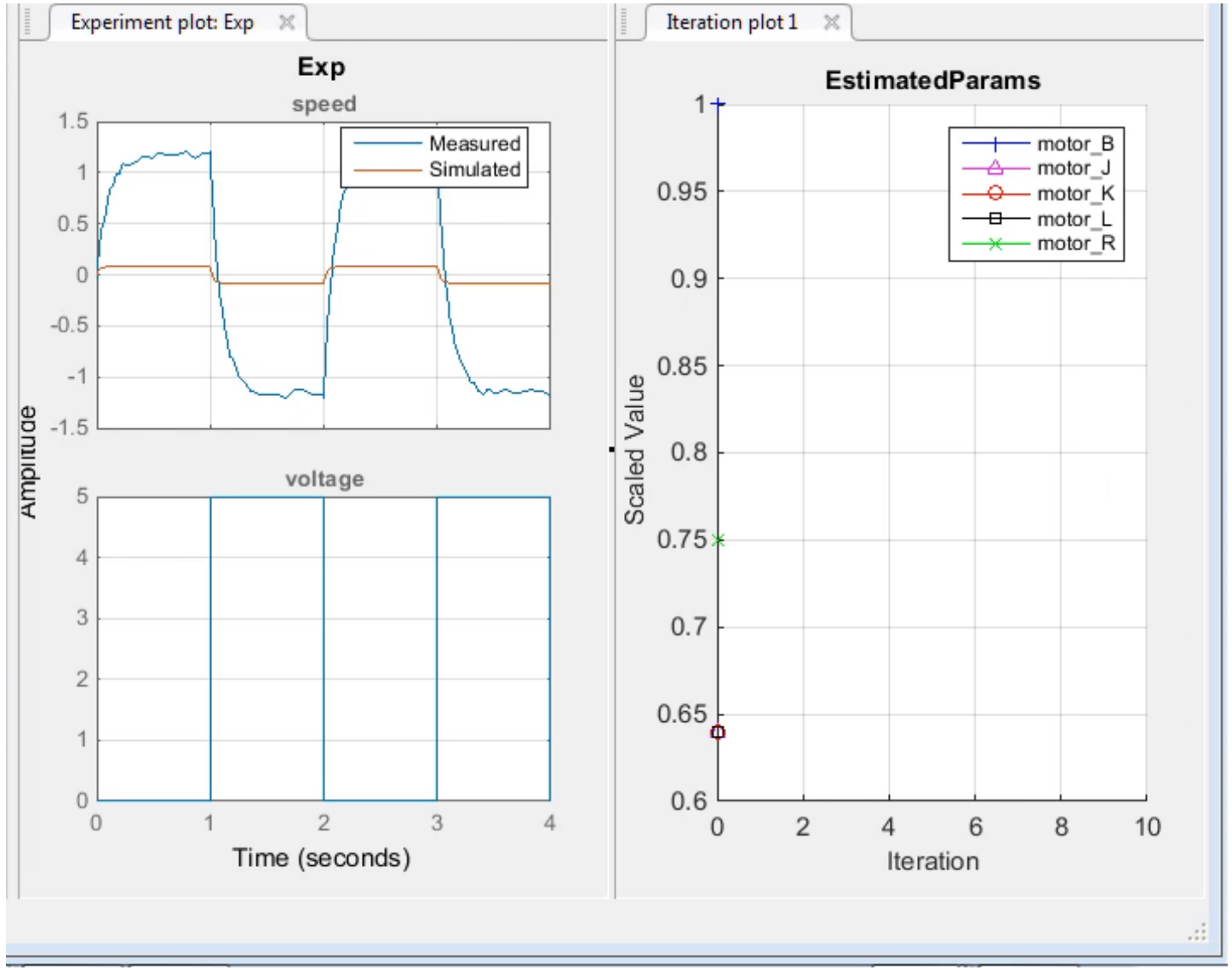


Speed run-down test

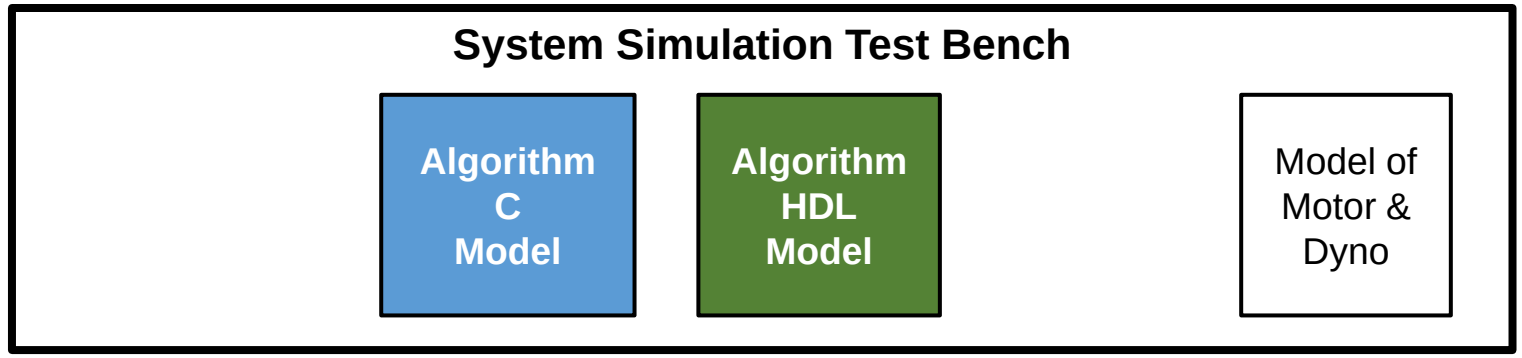
Inertia = Torque/gradient

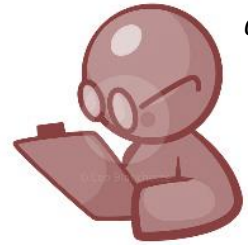
Estimating Parameters from measured data using Simulink Design Optimization

- Use simulation-based optimisation
  - match model parameters to real-world data



Adding Implementation Detail to Algorithms





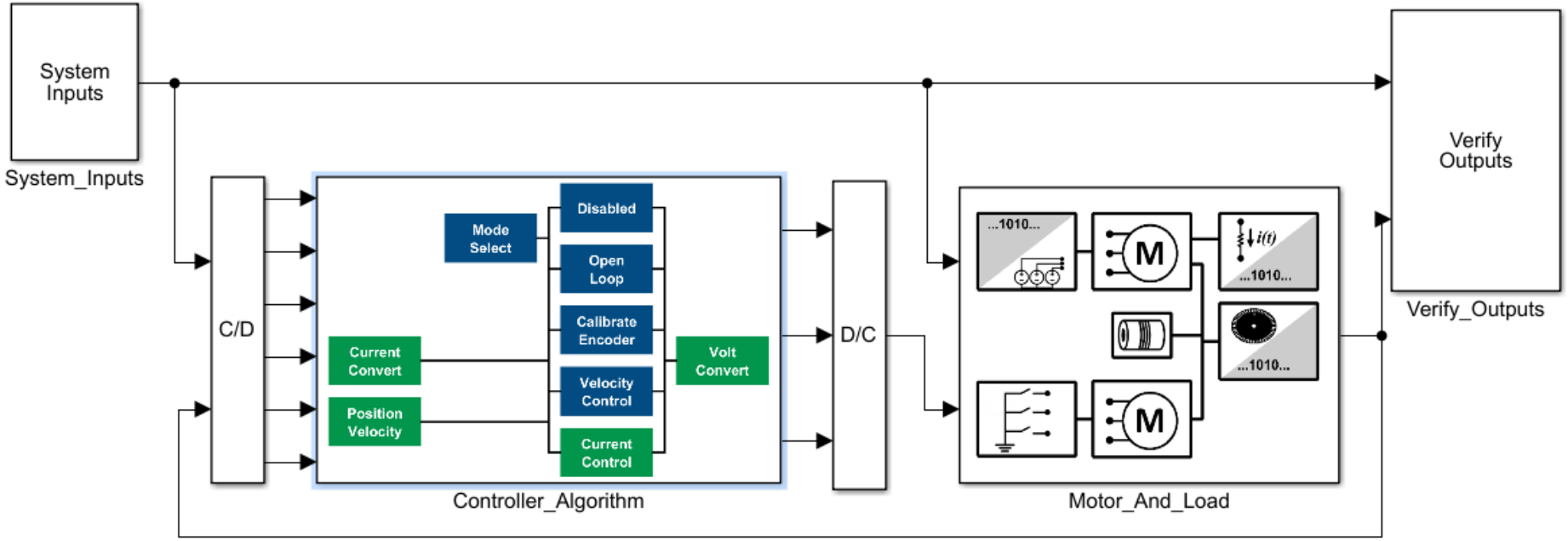
Algorithm developer

- Which parts of my algorithm should be implemented in C, and which in HDL?

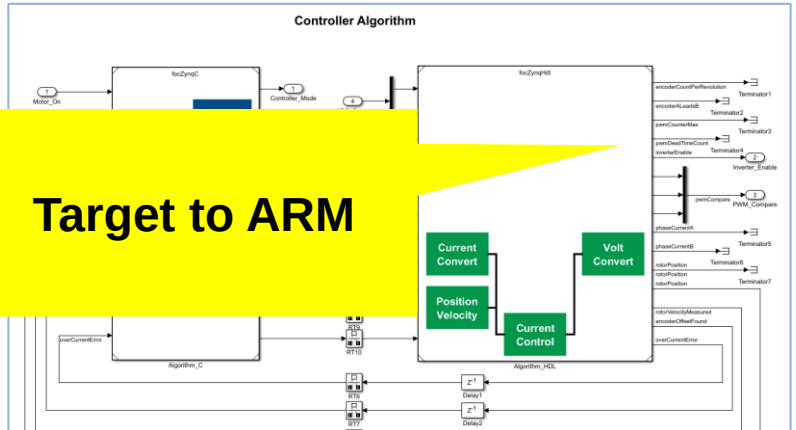
Strategies for Partitioning an Algorithm Between Hardware and Software

- Use experience
  - some timing requirements are known e.g. current control @25kHz
- Put everything on the software core and profile it
  - where are the bottlenecks?
  - Can these be moved to hardware?
- Put algorithms where the data comes in
  - minimise data transfer
- Monitor resource usage and move things when you are near the limit

Hardware/Software Partitioning



Target to Programmable Logic



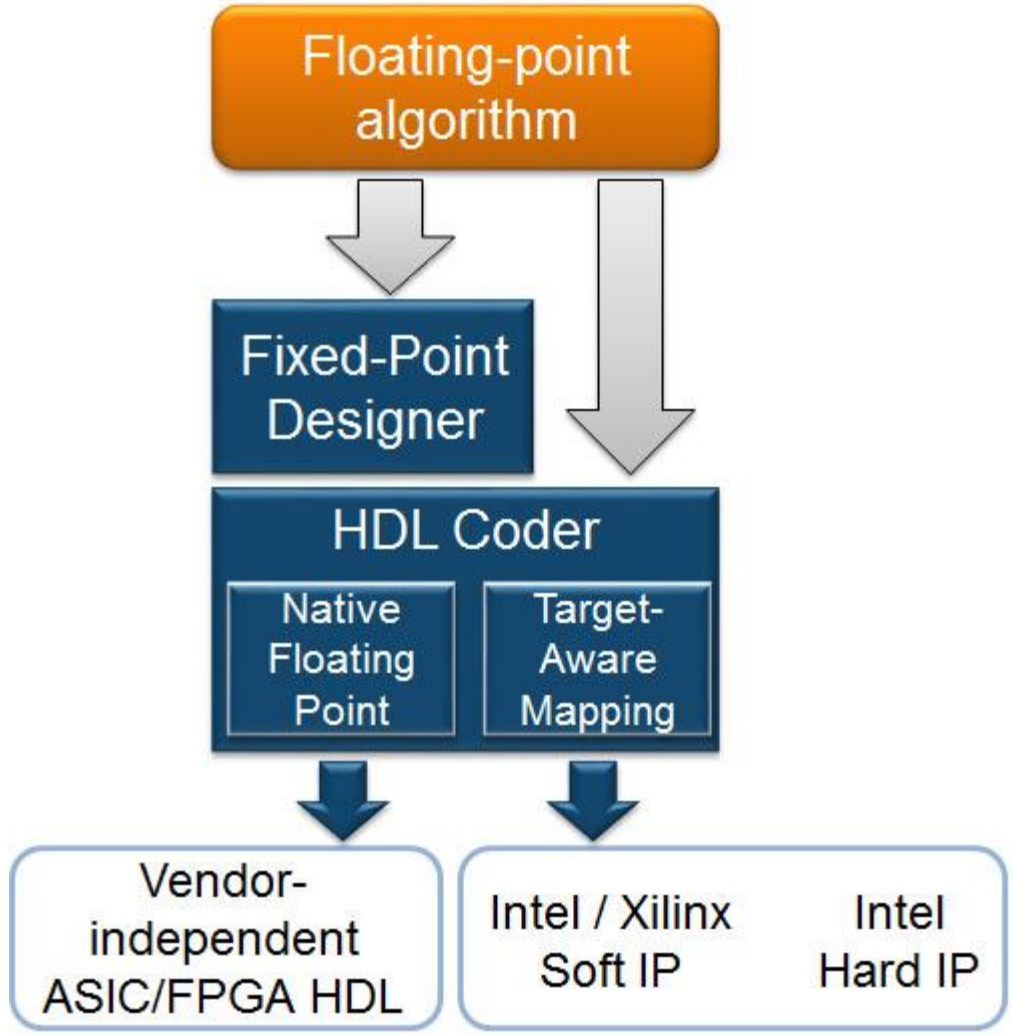
Target to ARM

Floating-point to fixed-point conversion

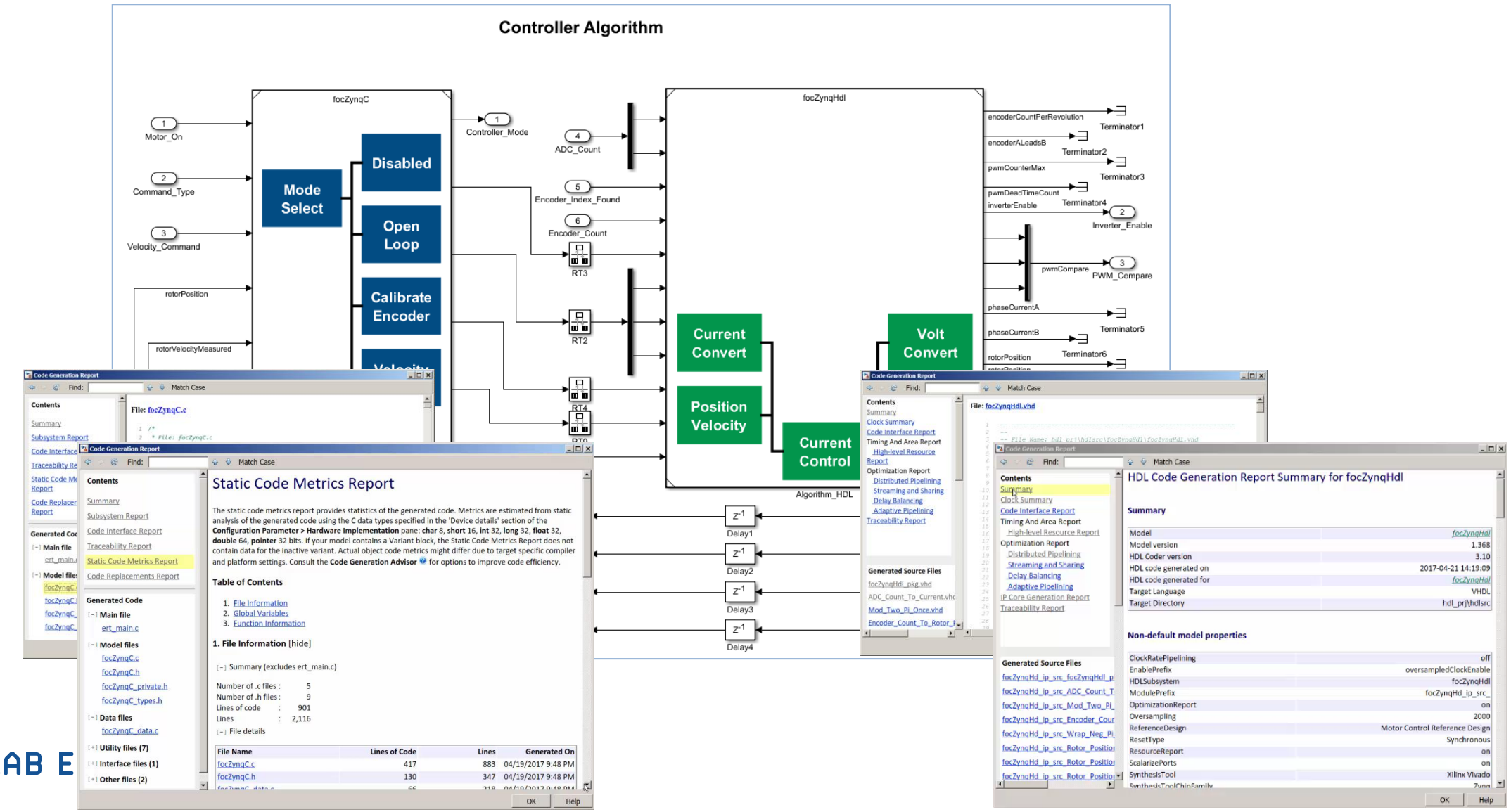
- Is it always necessary?
- Possibly, to meet resource constraints on the hardware

- Fixed-Point Designer™ helps automate the conversion process

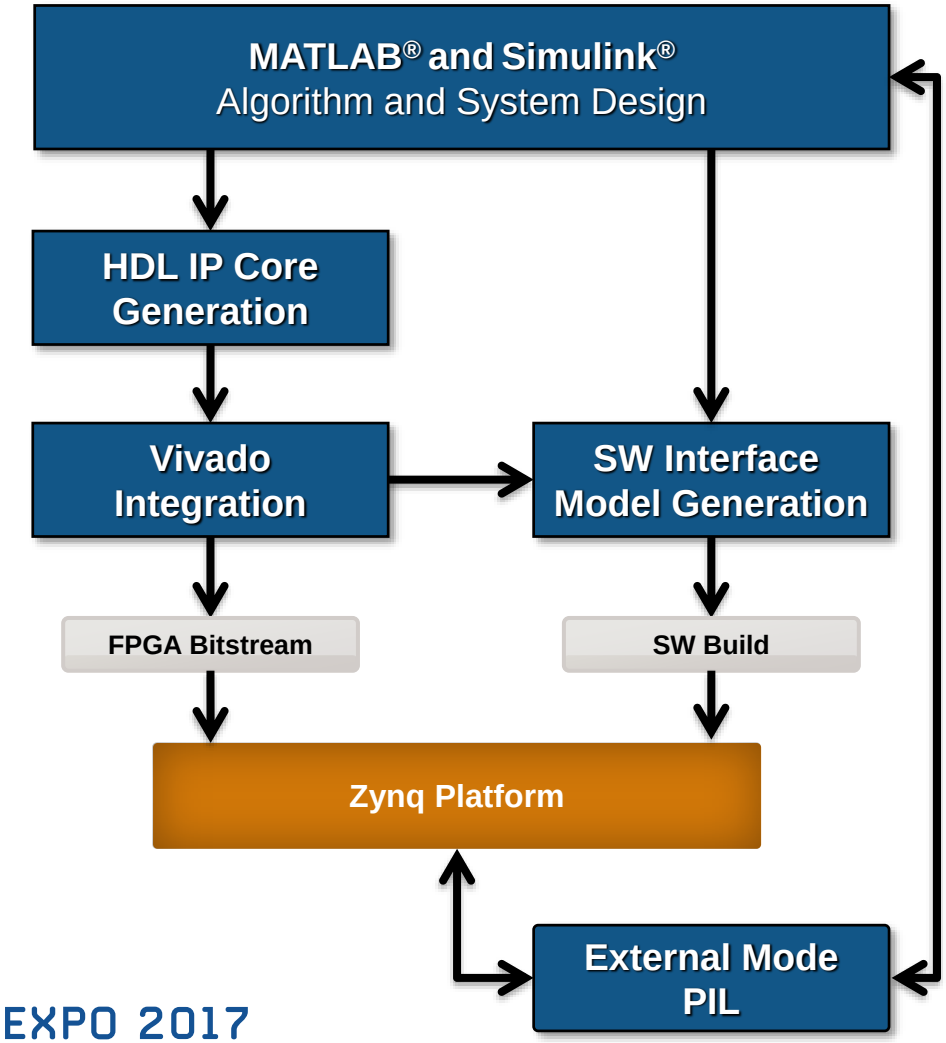
- HDL Coder™ native floating-point technology can generate HDL code from your floating-point design



Code Generation



Zynq Model-Based Design Workflow



- Real-time Parameter Tuning and Verification
  - External Mode
  - Processor-in-the-loop
- More probe and debug capability in the future

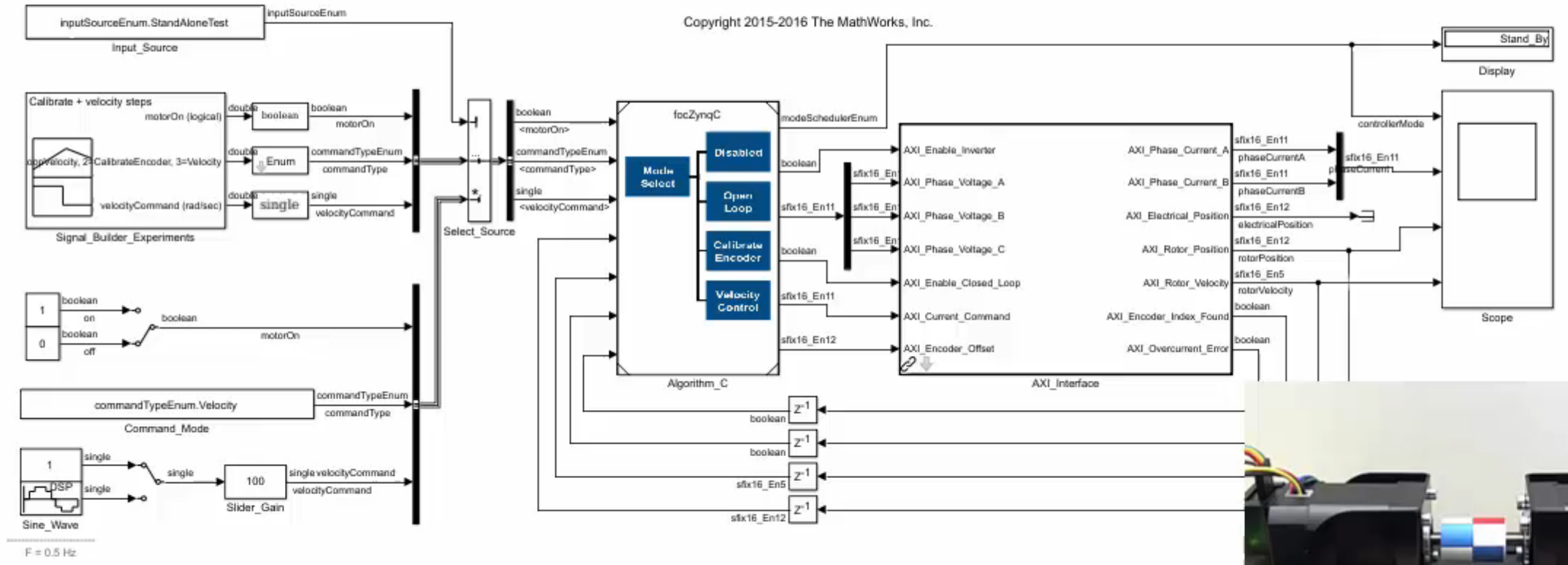
focZynqArmDeployment - Simulink

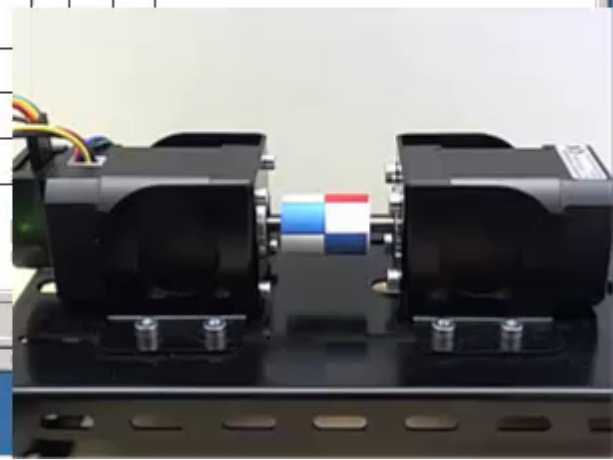
File Edit View Display Diagram Simulation Analysis Code Tools Help

focZynqArmDeployment

focZynqArmDeployment

Field-Oriented Control of Velocity
Zynq ARM Deployment for AD-FMCMOTCON2

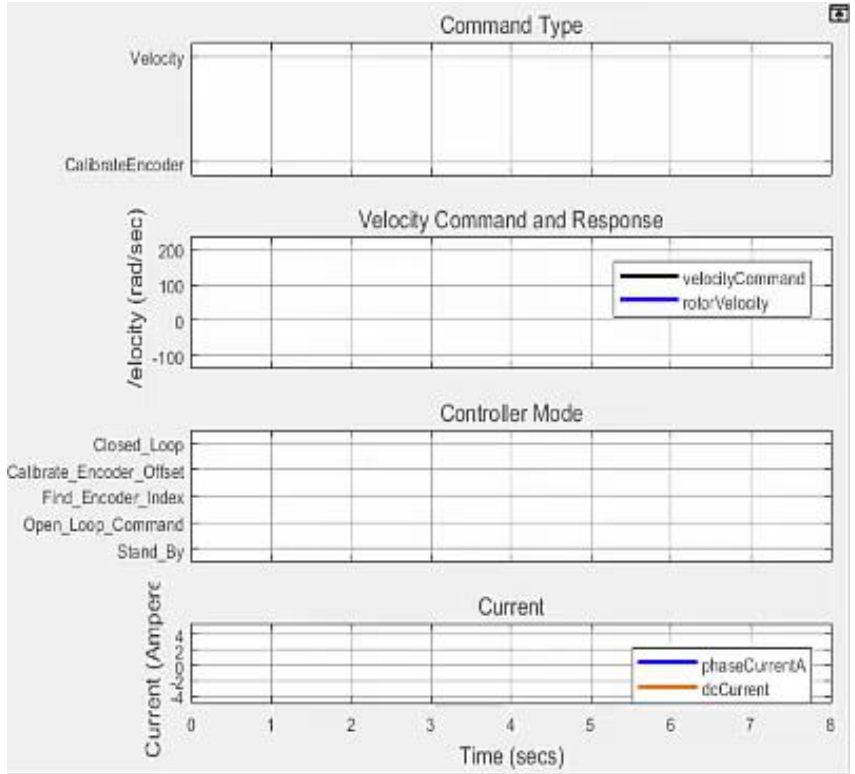


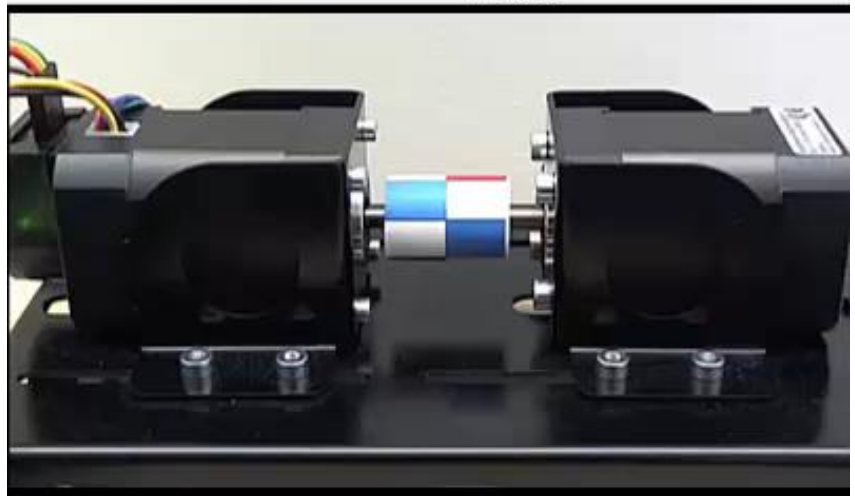


Ready

View diagnostics 61%

- Why use Hardware and Software for motor control?
- Why use Model-Based Design for motor control?
- How to use Model-Based Design for motor control?





Why use Model-Based Design to develop motor control applications?

- Enables early validation of specifications using simulation months before hardware is available.
- Dramatically improves design team collaboration and designer productivity by using a single design environment.
- Reduces hardware testing time by 5x by shifting design from lab to the desktop