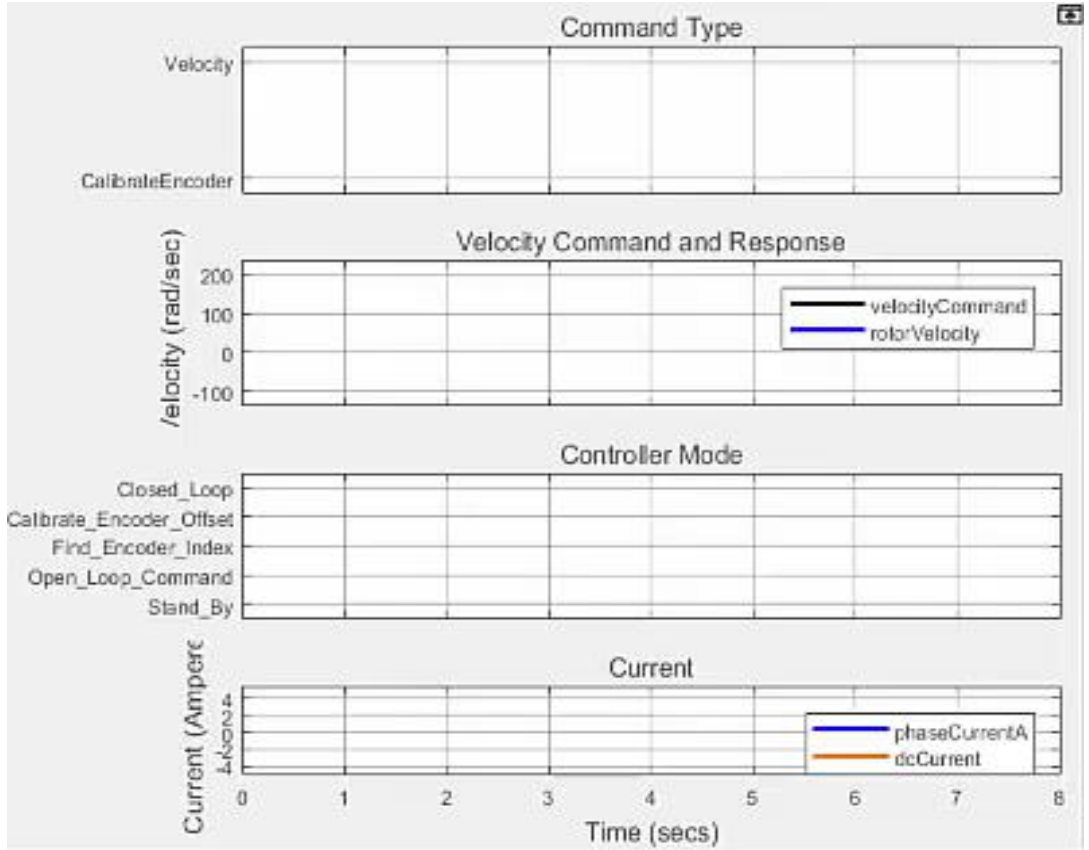
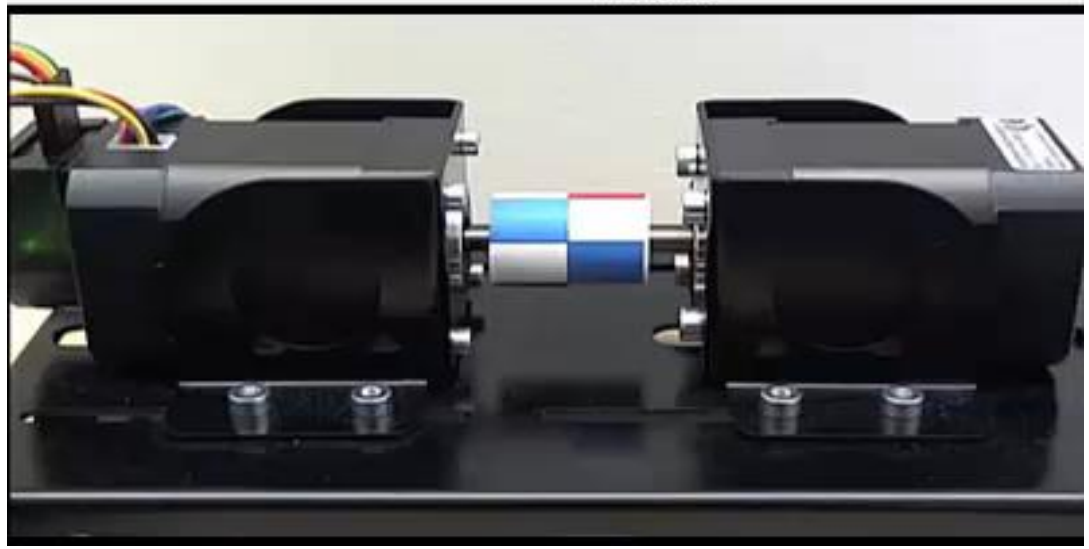
MATLAB EXPO 2018

Hardware and Software Co-Design for Motor Control Applications

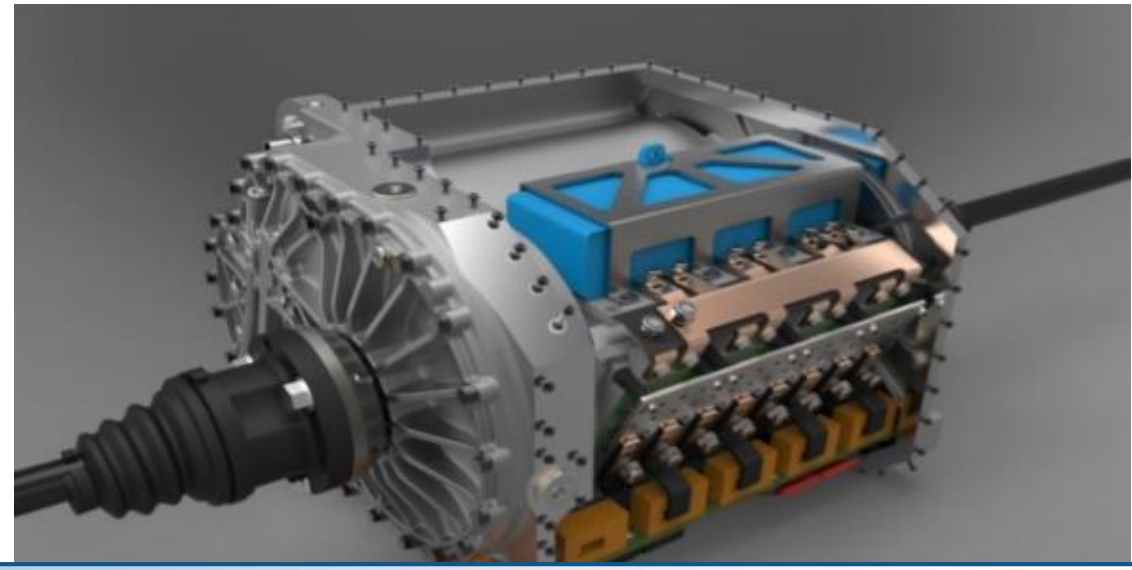
Stephan van Beek





Punch Powertrain develops complex SoC-based motor control

- Powertrains for hybrid and electric vehicles
- Hardware choice through simulations
- Traditional microcontroller too slow
- No experience designing FPGAs!

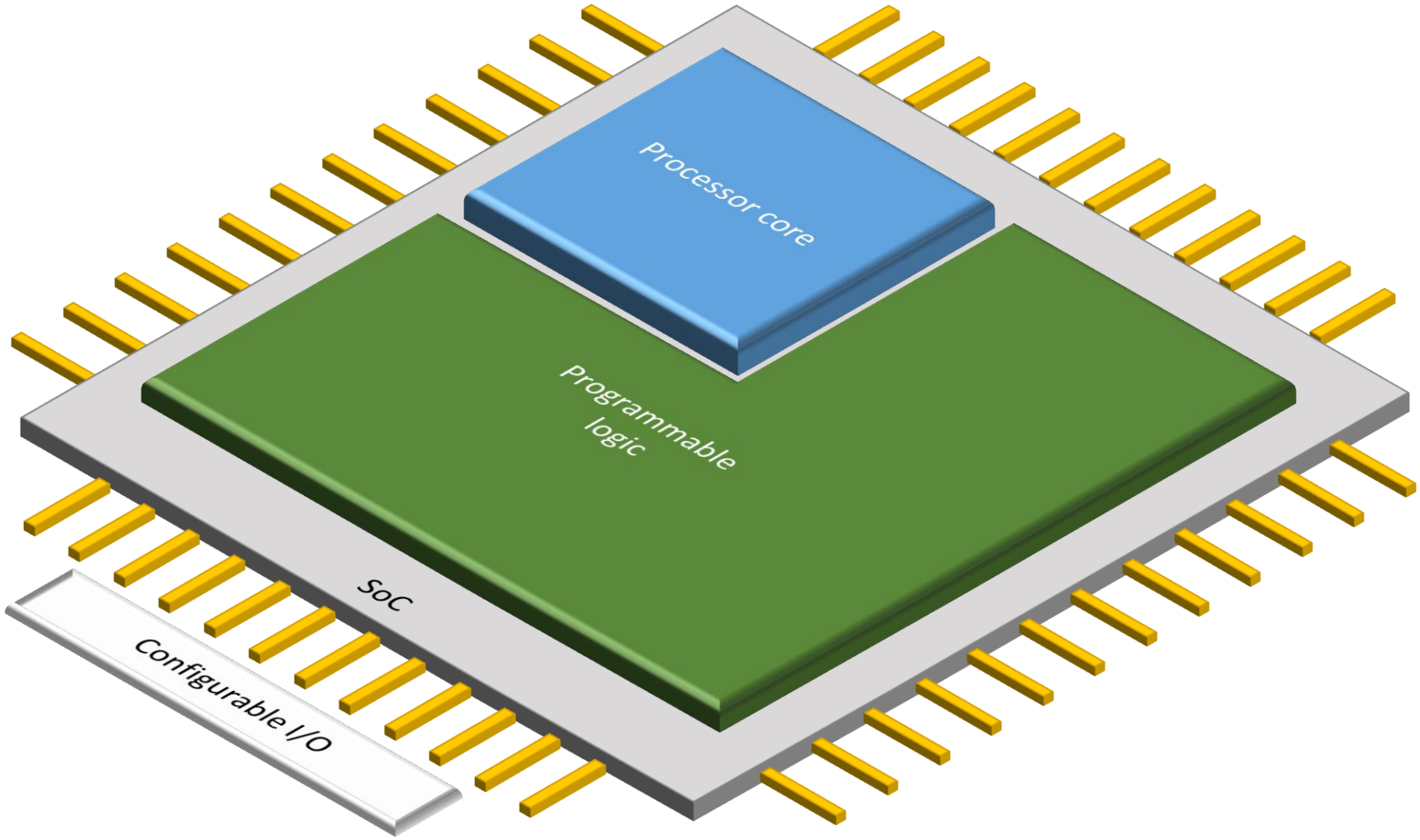


- ✓ Designed integrated E-drive: Motor, power electronics and software
- ✓ 4 different control strategies implemented
- ✓ Done in 1.5 years with 2FTE’s
- ✓ Models reusable for production
- ✓ Smooth integration and validation due to development process

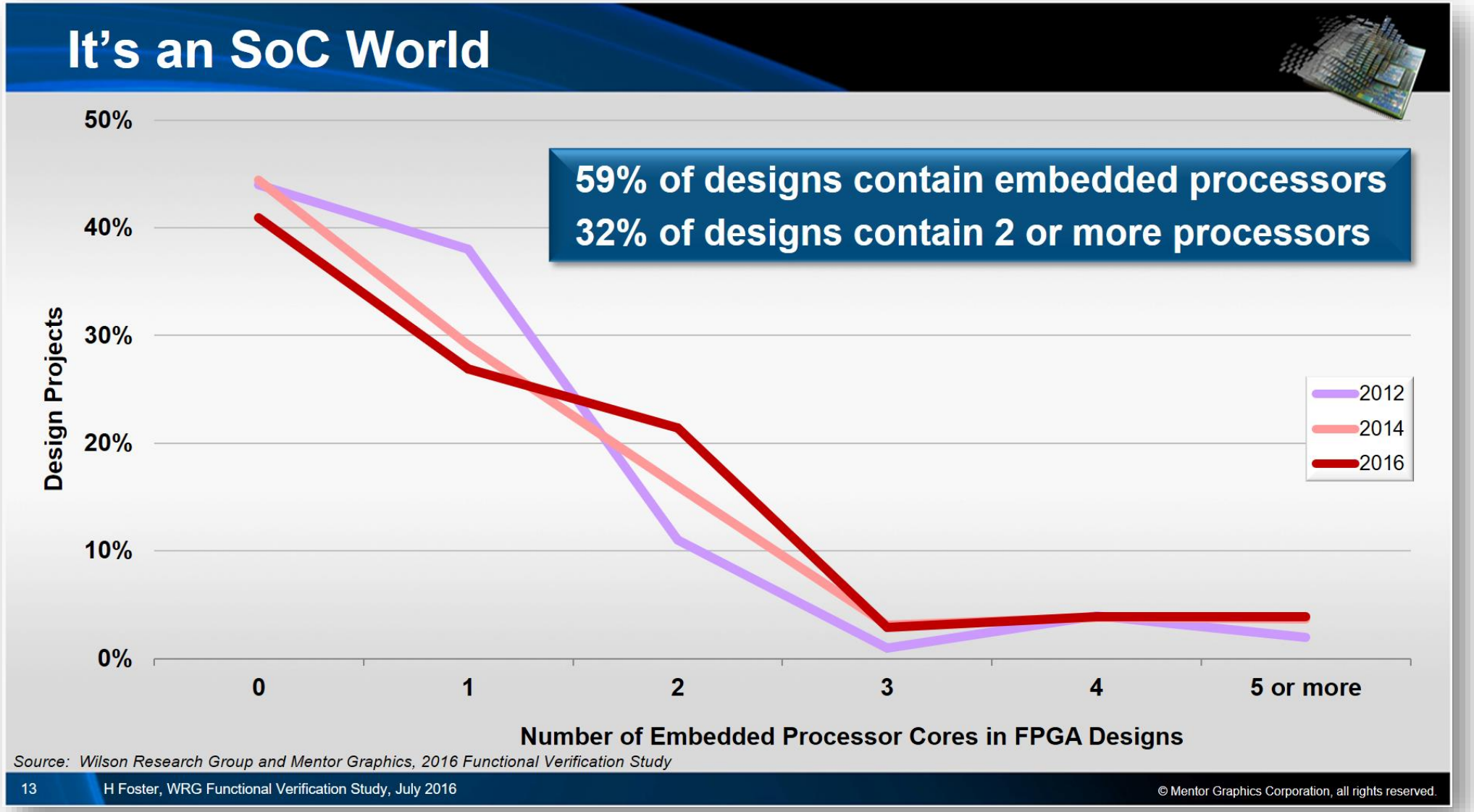
Link to video

Key trend: Increasing demands from motor drives

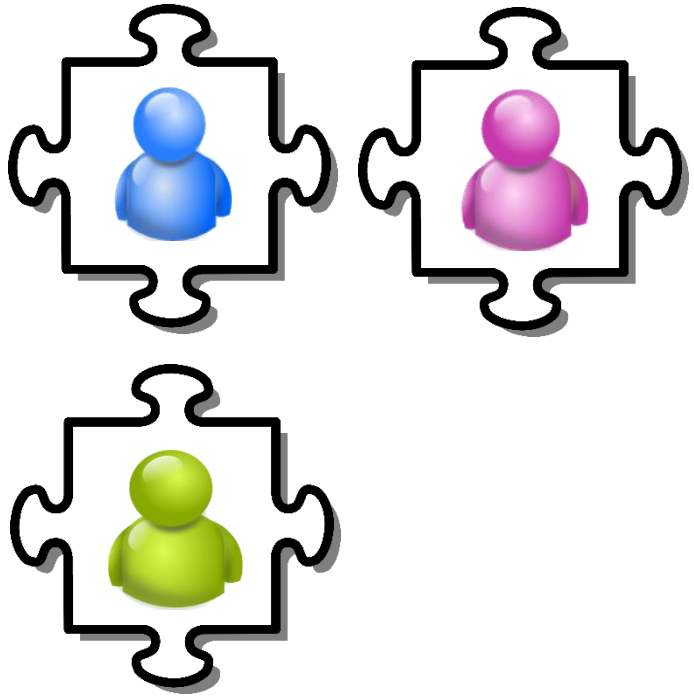
Systems-on-Chip for motor and power control

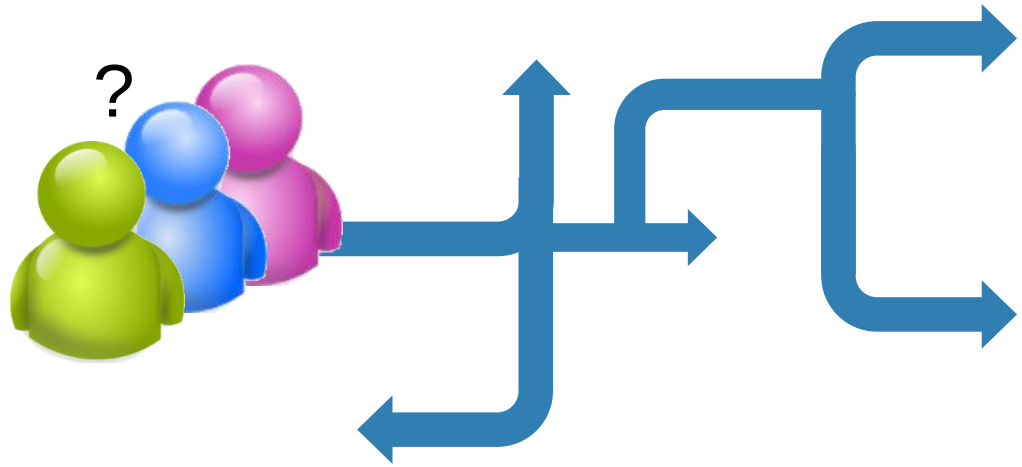


Key Trend: ~60% FPGA projects contain embedded processors

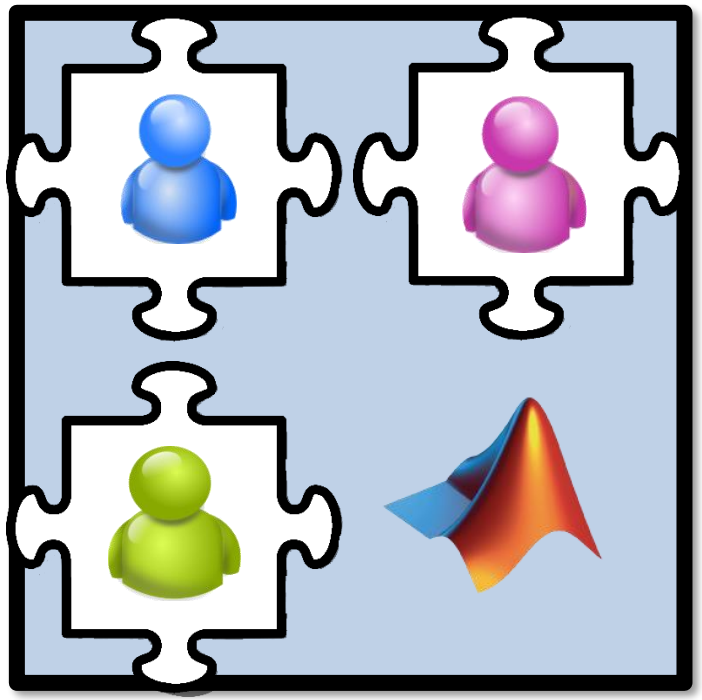


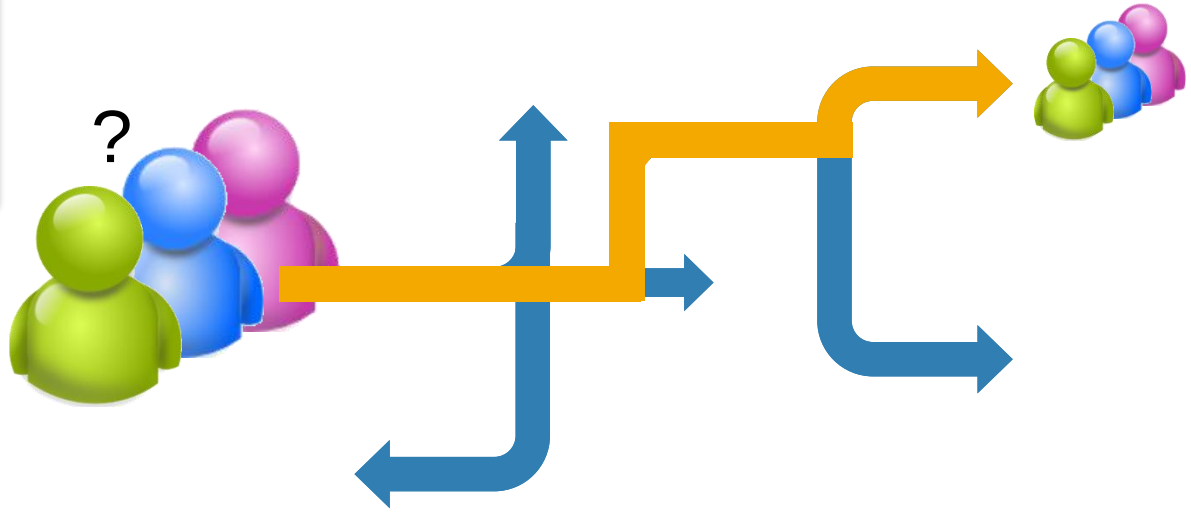
Challenges in using SoCs for motor and power control





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



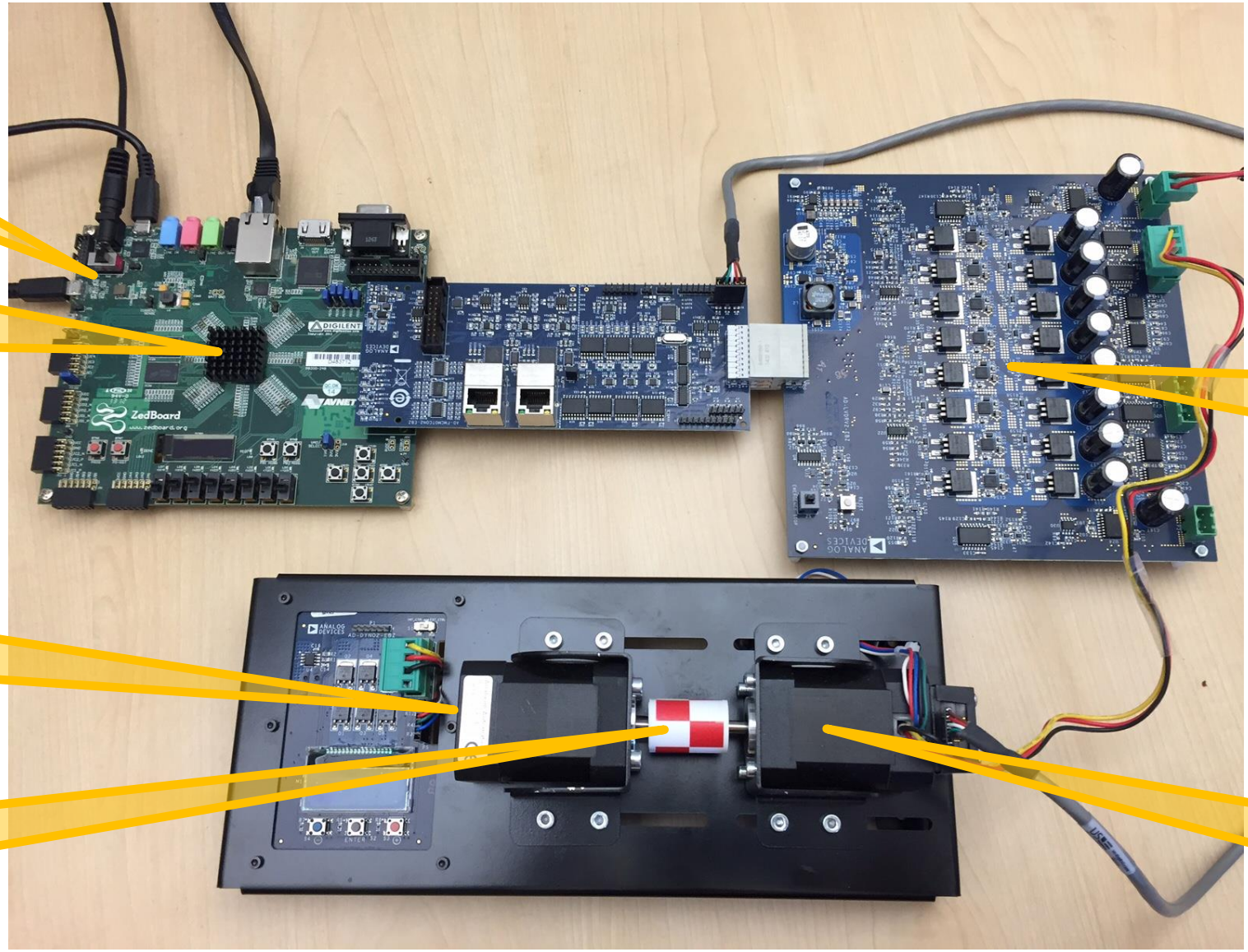


ZedBoard

Zynq SoC
(XC7Z020)

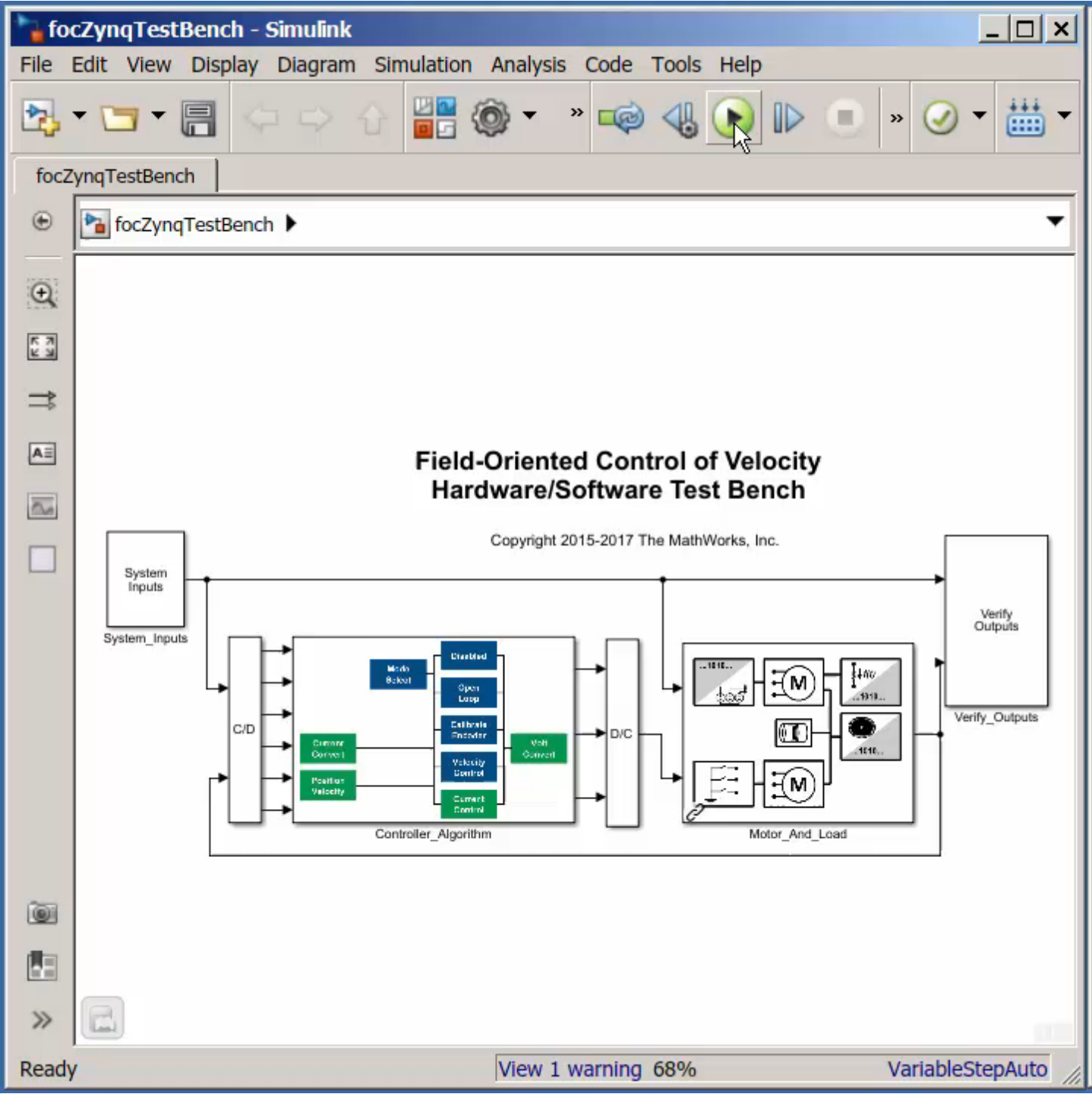
Load motor

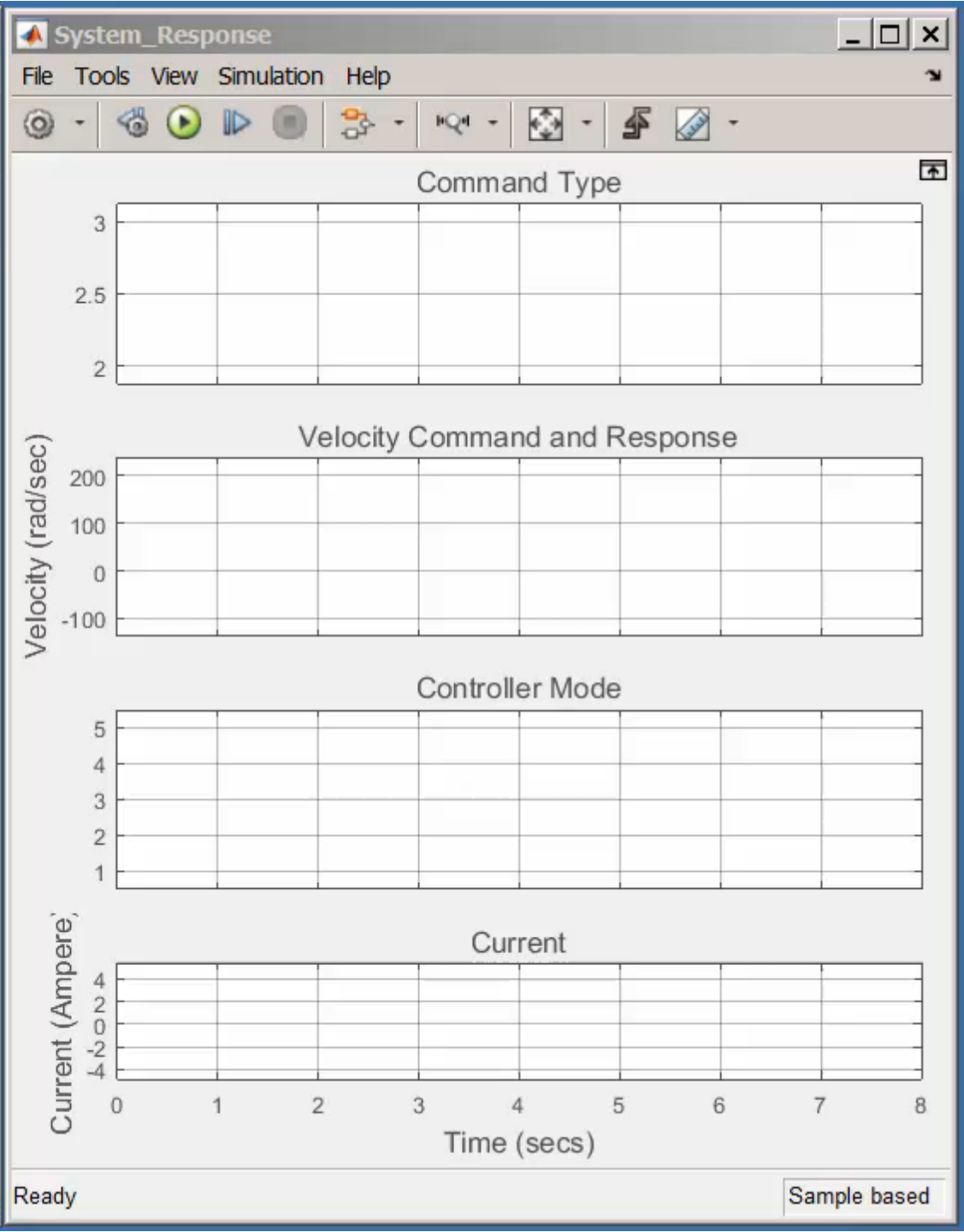
Mechanical coupler



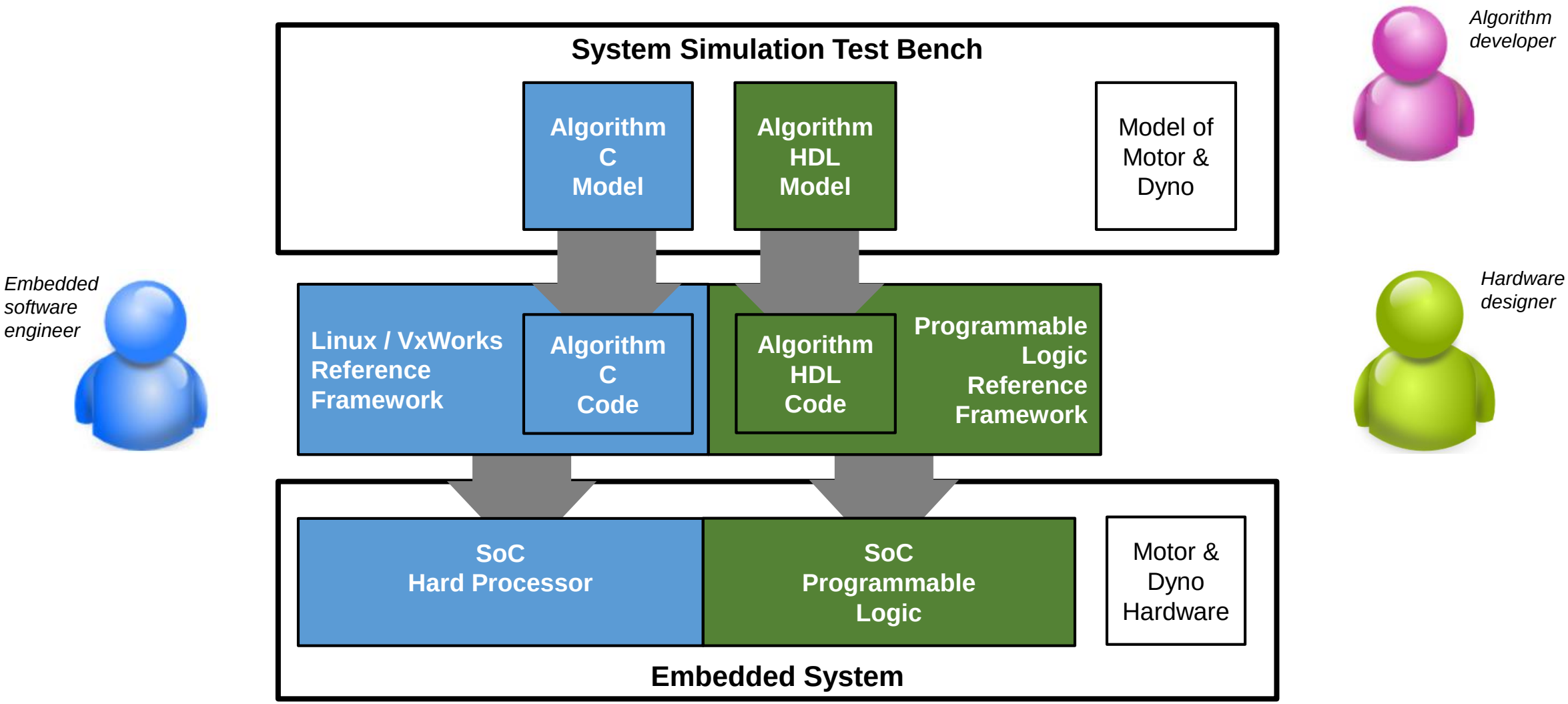
FMC module: control board + low-voltage board

Motor under test (with encoder)

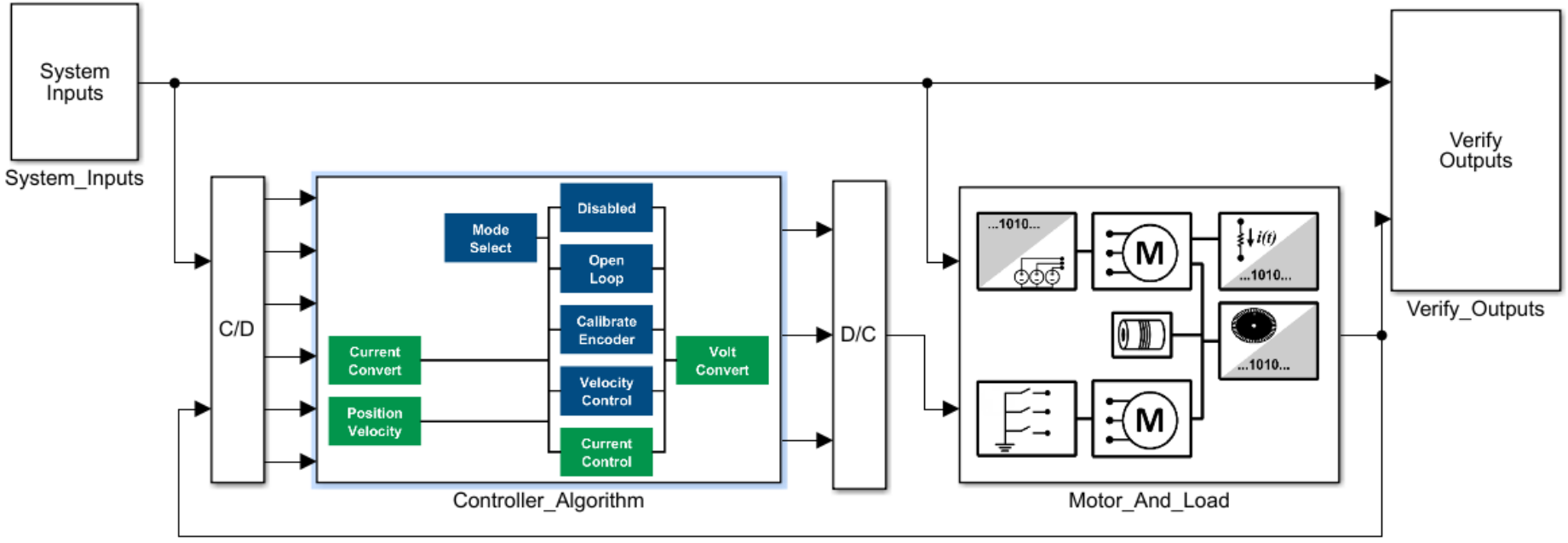

focZynqTestBench - Simulink
File Edit View Display Diagram Simulation Analysis Code Tools Help
focZynqTestBench
focZynqTestBench
Field-Oriented Control of Velocity
Hardware/Software Test Bench
Copyright 2015-2017 The MathWorks, Inc.
System Inputs
System_Inputs
C/D
Controller_Algorithm
D/C
Motor_And_Load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

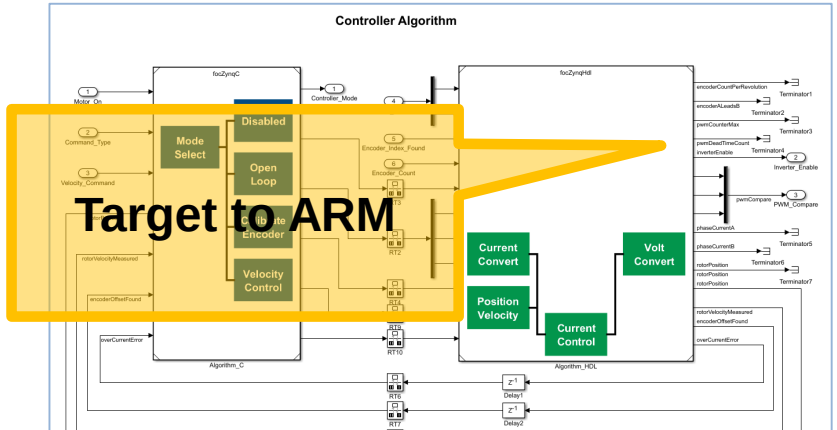
Conceptual workflow targeting SoCs



Hardware/software partitioning

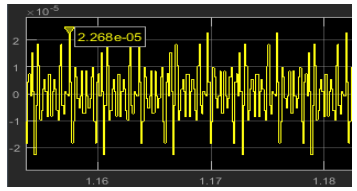


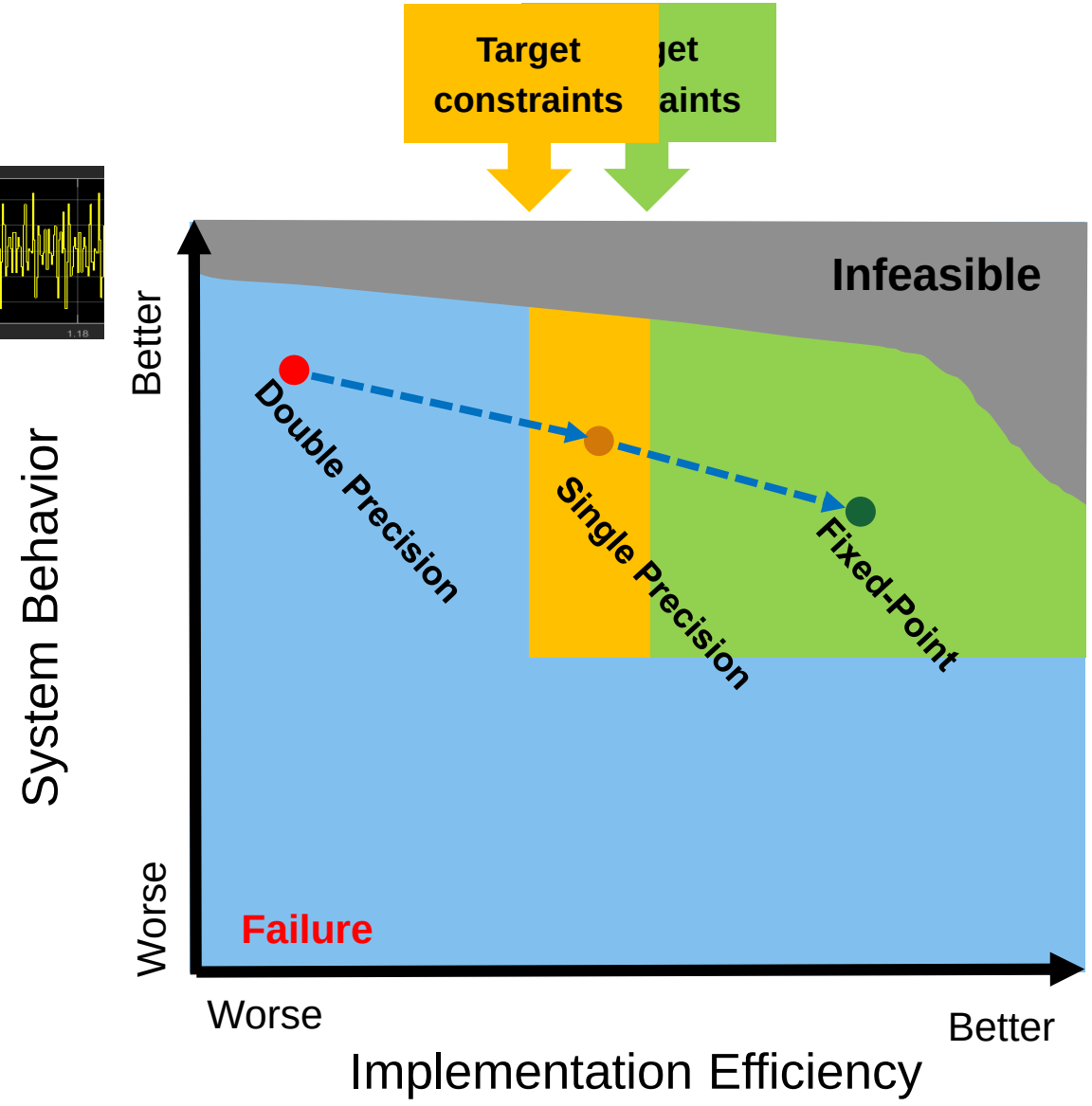
Target to Programmable Logic



Target to ARM

Efficient embedded designs

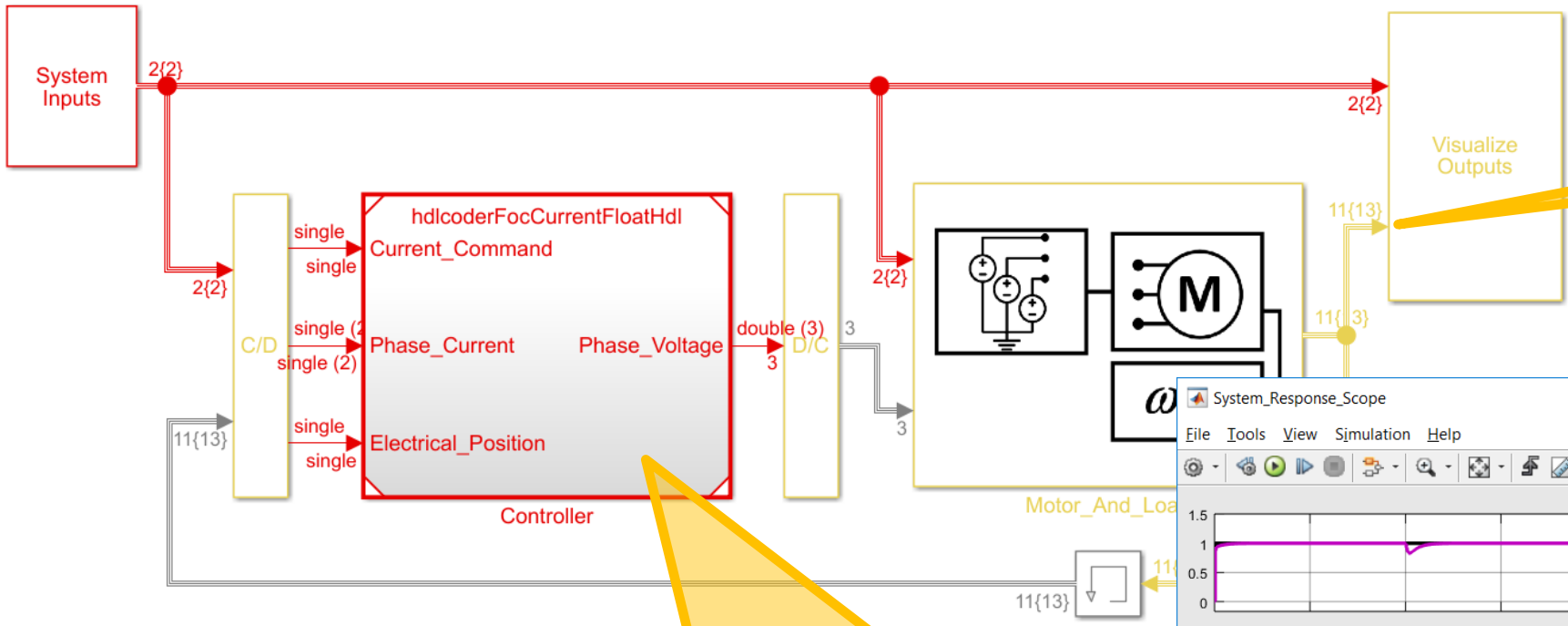




Considerations

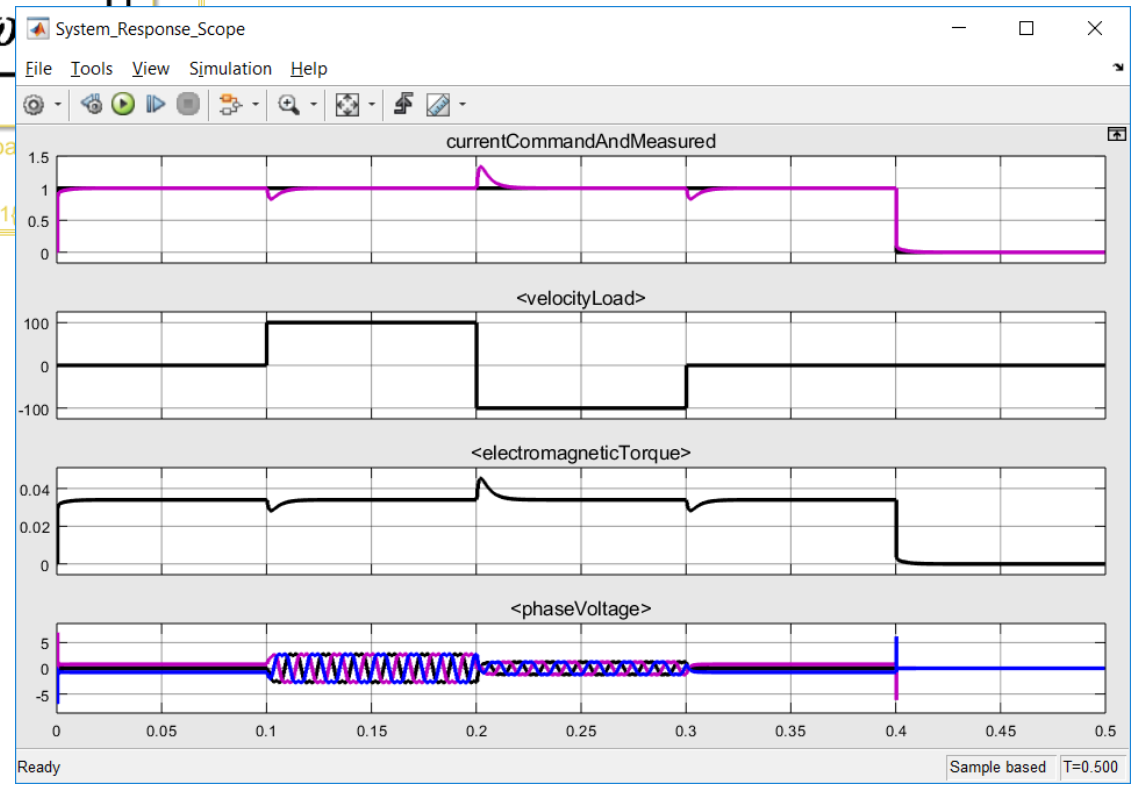
- Design Time
- Resource usage
- Flexibility
- Ease of design
- ….

Floating point to Fixed-Point made easy

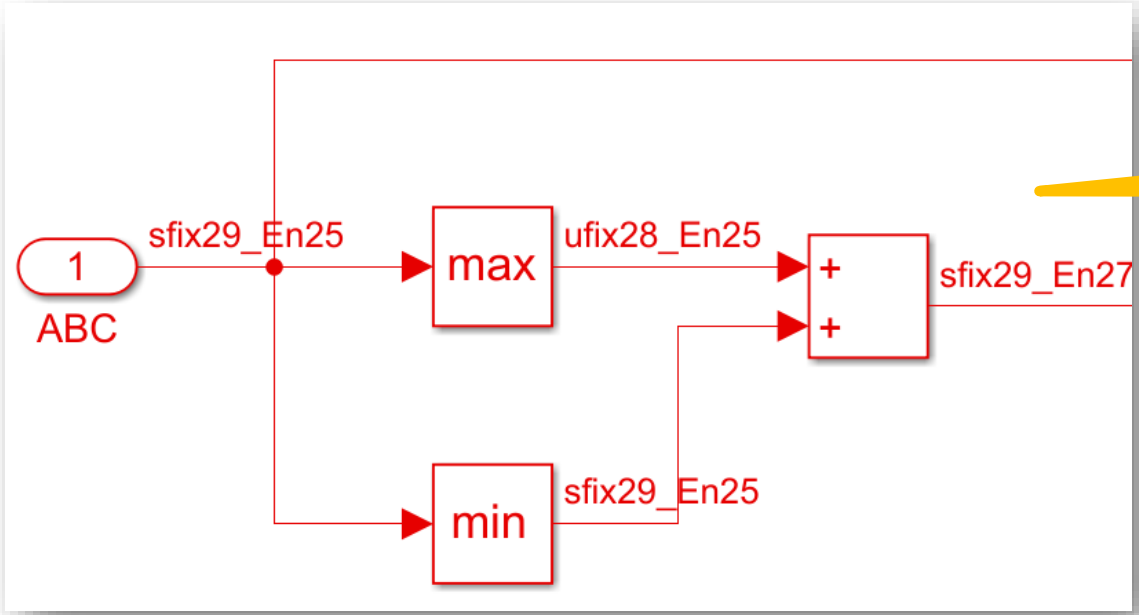


Tolerance specification to guide Fixed-Point conversion

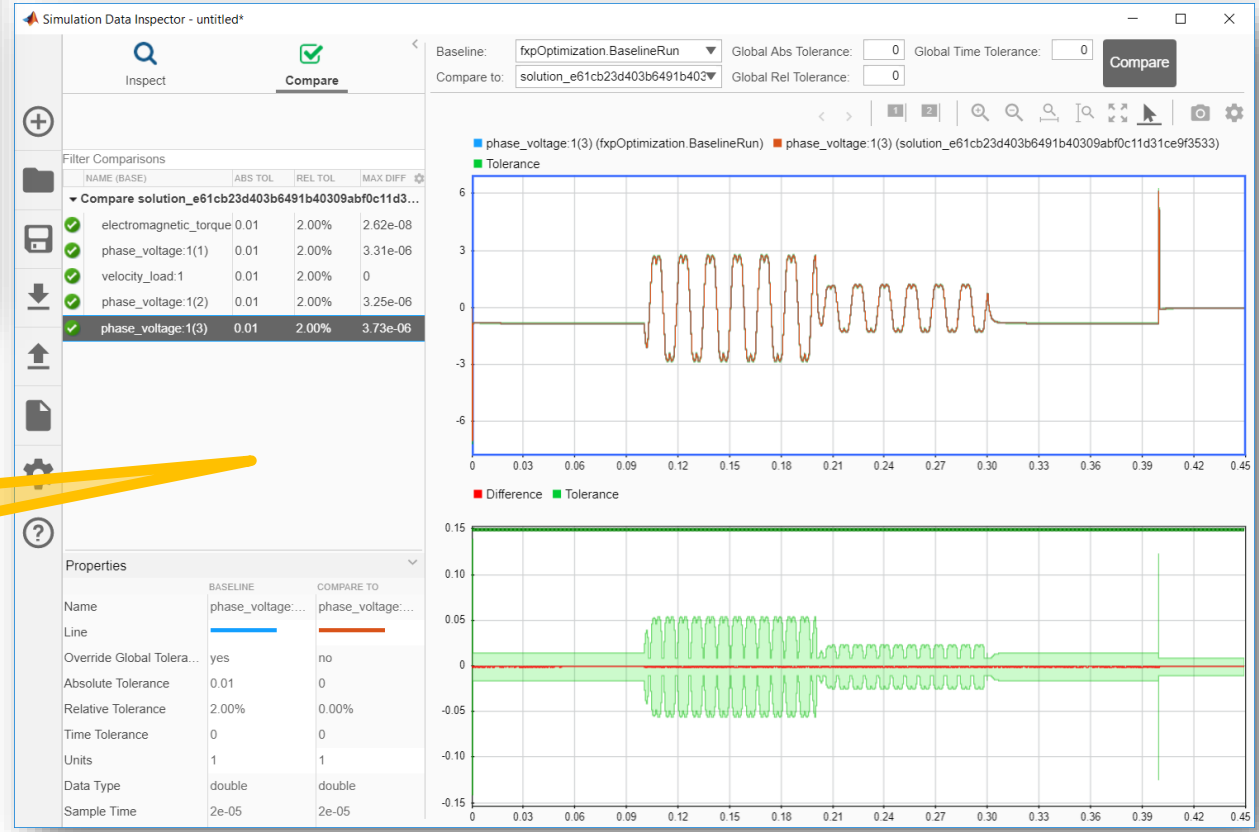
Convert to Fixed-Point at lowest implementation cost but still meeting system level specifications



Automatic Fixed-Point optimization

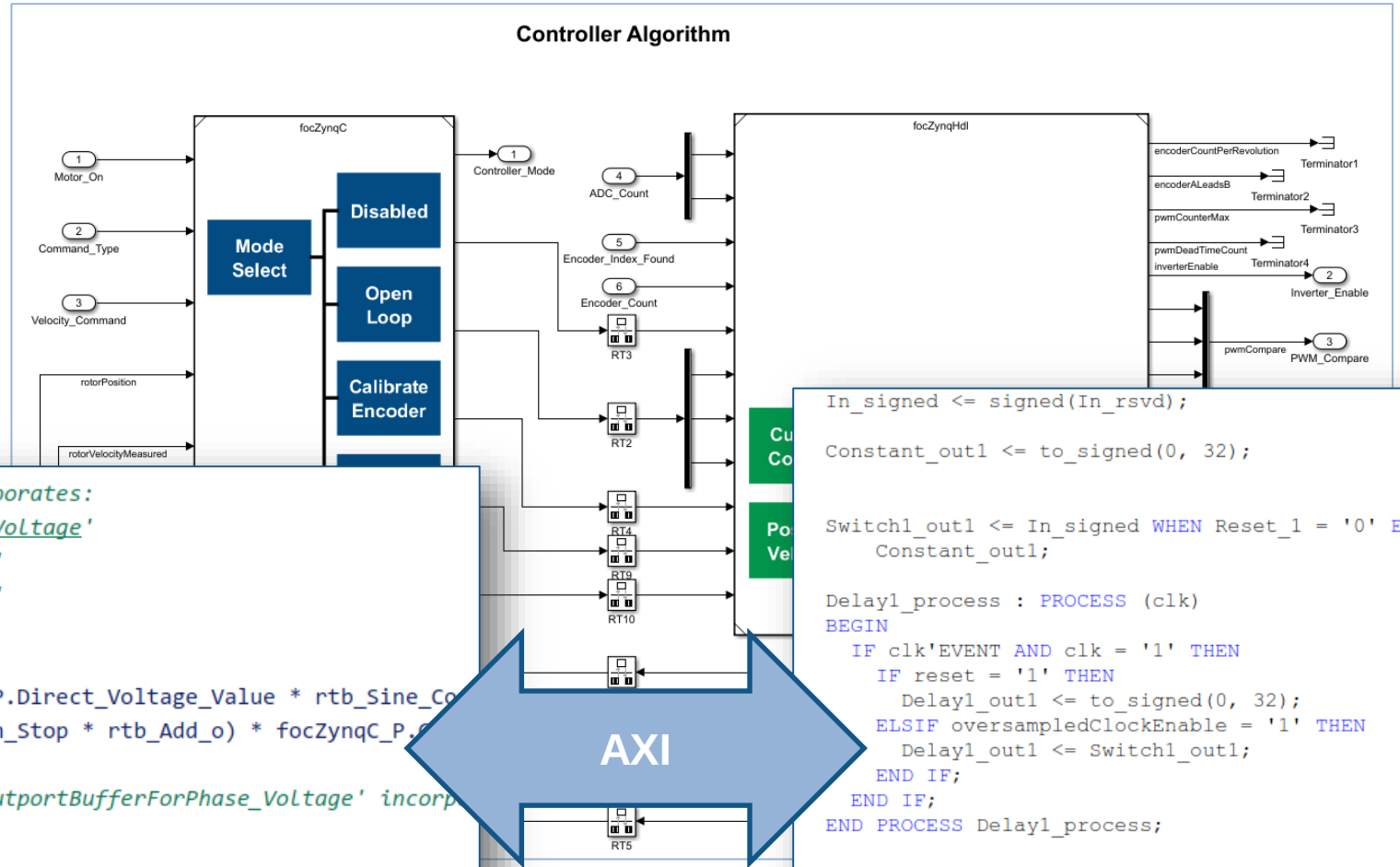


Trade-off Fixed-Point versus Floating Point types



Automatic validation of compliance to tolerance specifications

Code Generation




```
/* Gain: '<S12>/Gain1' incorporates:
 *  Constant: '<S10>/Direct_Voltage'
 *  Product: '<S13>/Product3'
 *  Product: '<S13>/Product4'
 *  Sum: '<S13>/Add2'
 */
rtb_Switch_Stop = (focZynqC_P.Direct_Voltage_Value * rtb_Sine_Co
                   rtb_Switch_Stop * rtb_Add_o) * focZynqC_P.

/* SignalConversion: '<S3>/OutportBufferForPhase_Voltage' incorp
 *  Sum: '<S12>/Add'
 *  Sum: '<S12>/Add1'
 */
focZynqC_B.Merge[1] = rtb_Switch_Stop - rtb_Difference;
focZynqC_B.Merge[2] = (0.0F - rtb_Difference) - rtb_Switch_Stop;
```

```
In_signed <= signed(In_rsvd);

Constant_out1 <= to_signed(0, 32);


Switch1_out1 <= In_signed WHEN Reset_1 = '0' ELSE
    Constant_out1;

Delay1_process : PROCESS (clk)
BEGIN
  IF clk'EVENT AND clk = '1' THEN
    IF reset = '1' THEN
      Delay1_out1 <= to_signed(0, 32);
    ELSIF oversampledClockEnable = '1' THEN
      Delay1_out1 <= Switch1_out1;
    END IF;
  END IF;
END PROCESS Delay1_process;



Switch_out1 <= Delay1_out1 WHEN Reset_1 = '0' ELSE
    Constant_out1;

Out_rsvd <= std_logic_vector(Switch_out1);
```

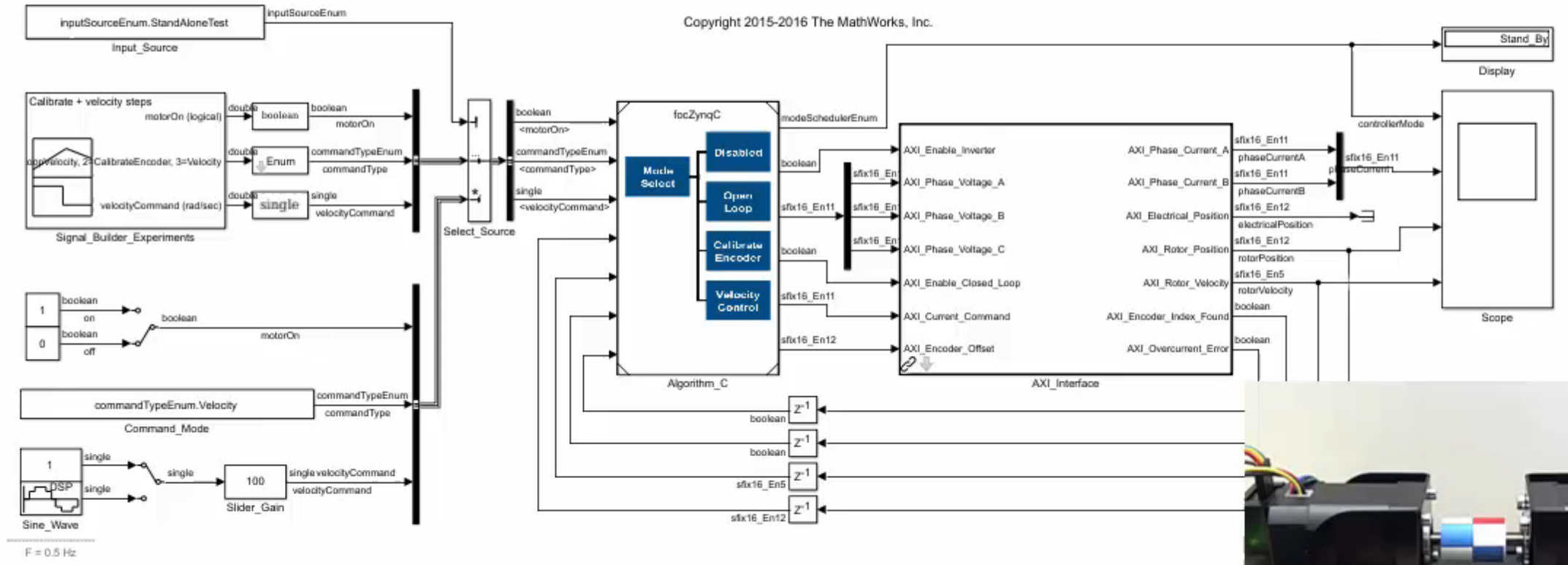
focZynqArmDeployment - Simulink

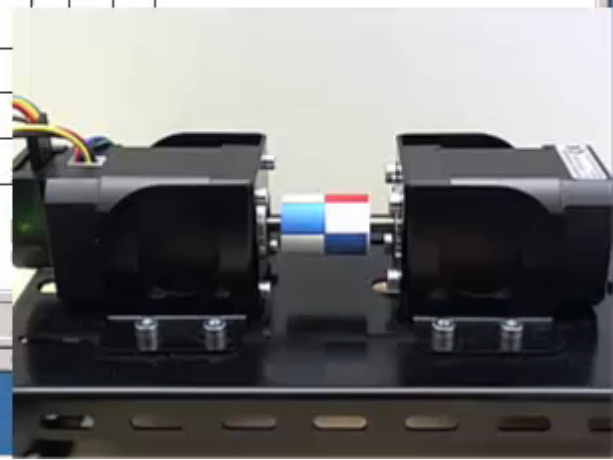
File Edit View Display Diagram Simulation Analysis Code Tools Help

focZynqArmDeployment

focZynqArmDeployment

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

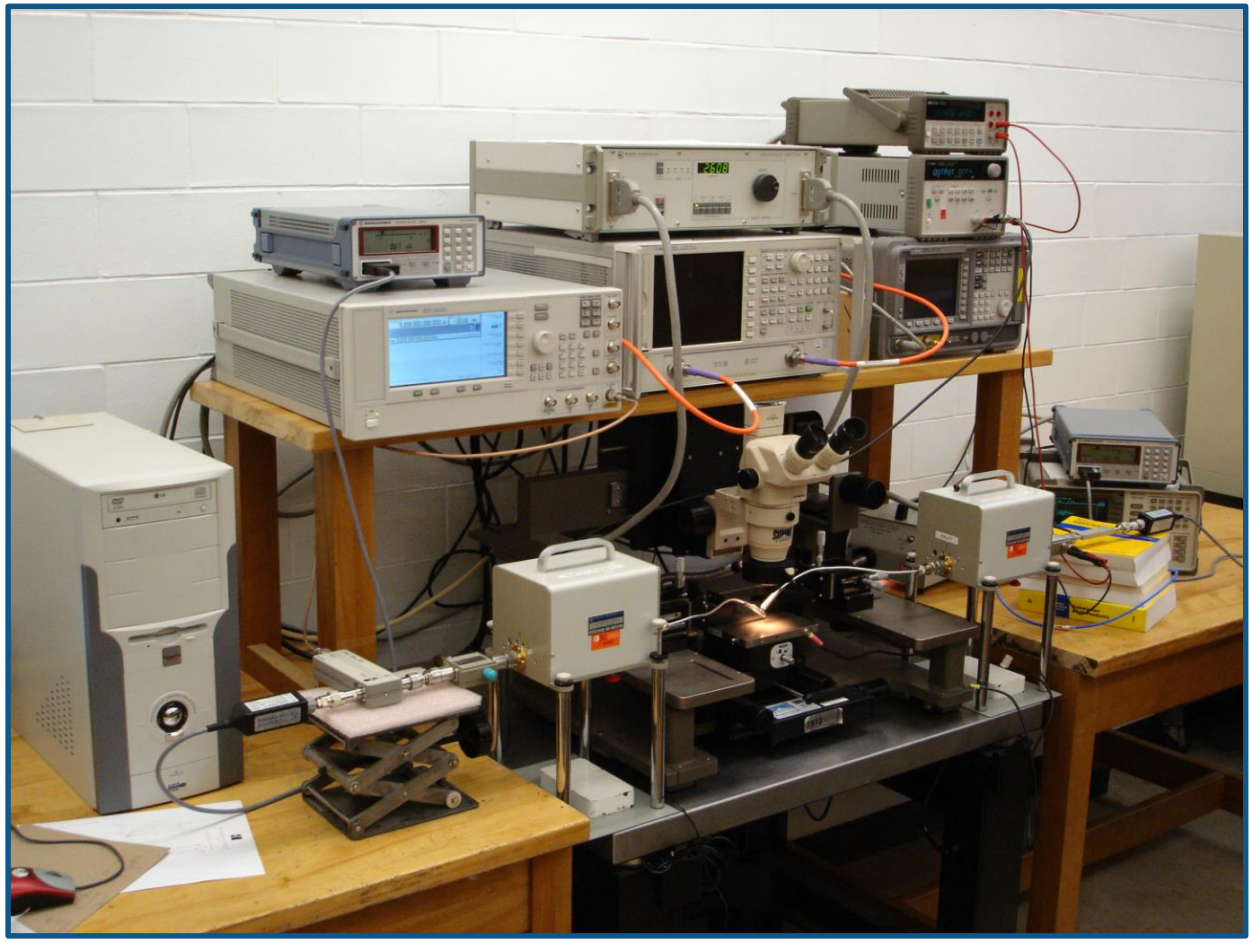




Ready

View diagnostics 61%

How are you going to debug your FPGA designs?

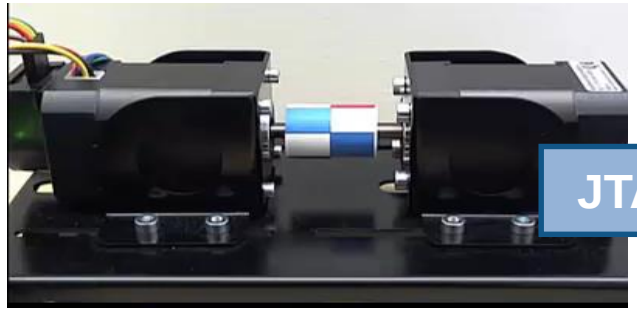


Some of the things you have to worry about:

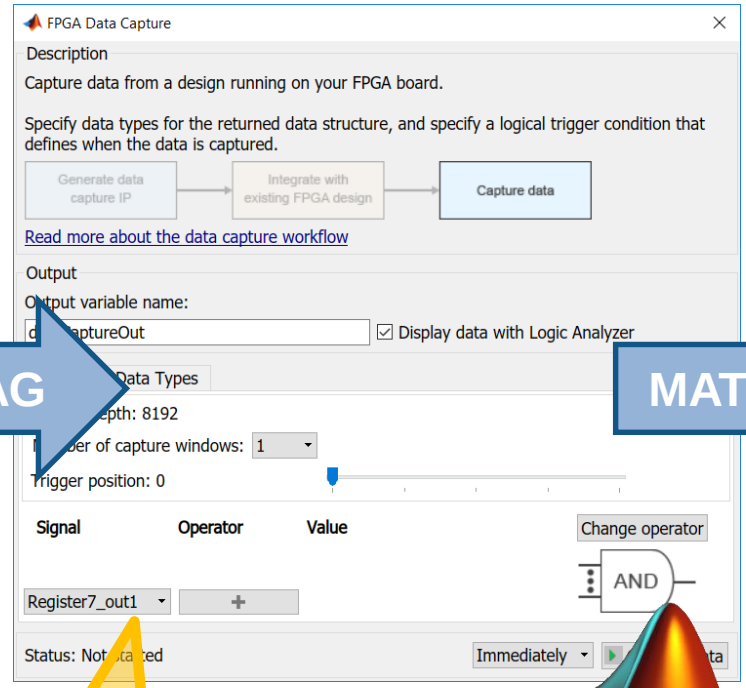
- How to capture high-rate or internal signals
- How to analyse my data?
- Can I automate this?
- Are my measurements reproducible?

Integrate debugging with MATLAB

FPGA Data Capture

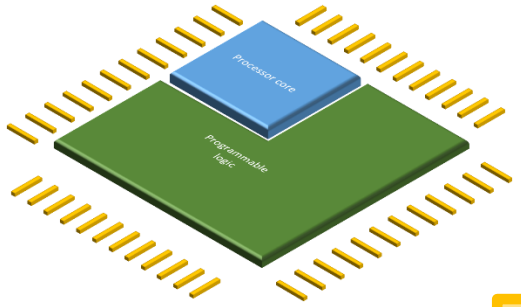


JTAG



MATLAB

Automating the analysis of large datasets



Easy access to internal FPGA signals

Reproducing real-world scenarios in simulation

3T Develops Robot Emergency Braking System with Model-Based Design

Challenge

Design and implement a robot emergency braking system with minimal hardware testing

Solution

Model-Based Design with Simulink and HDL Coder to model, verify, and implement the controller

Results

- Cleanroom time reduced from weeks to days
- Late requirement changes rapidly implemented
- Complex bug resolved in one day

A SCARA robot.

"With Simulink and HDL Coder we eliminated programming errors and automated delay balancing, pipelining, and other tedious and error-prone tasks. As a result, we were able to easily and quickly implement change requests from our customer and reduce time-to-market."

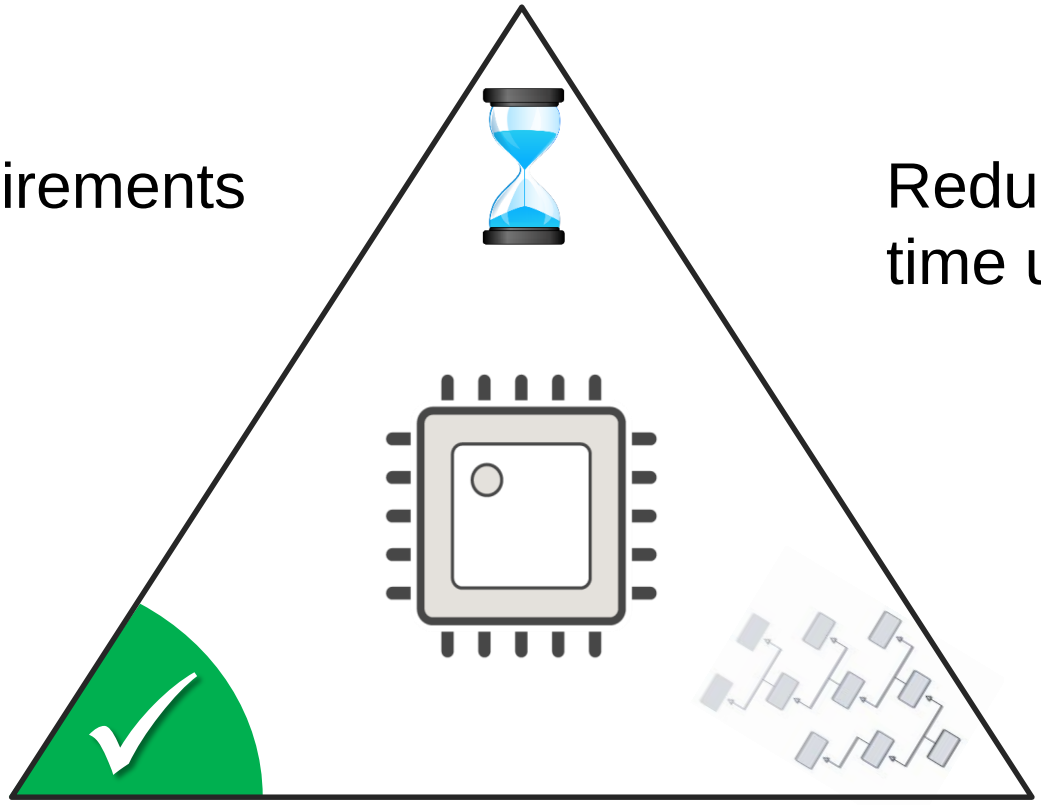
Ronald van der Meer

3T

Link to user story

Key Takeaways

Meet stringent requirements and reduce costs



Reduce hardware testing time up to 5x

Manage design complexity and improve team collaboration

How to get started?

- Embedded Systems
- DSP for FPGA Design
- Xilinx Zynq SoCs