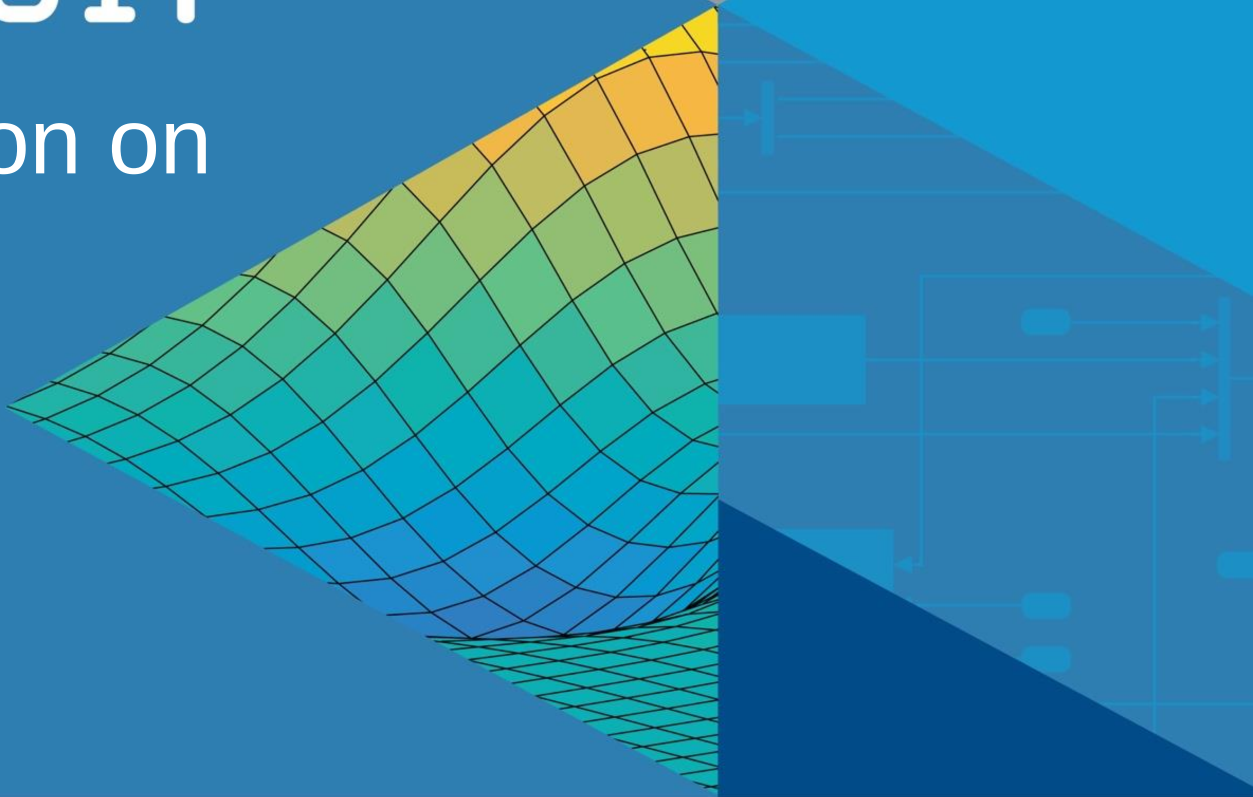
MATLAB EXPO 2017

Motor Controls Implementation on Systems-On-Chip

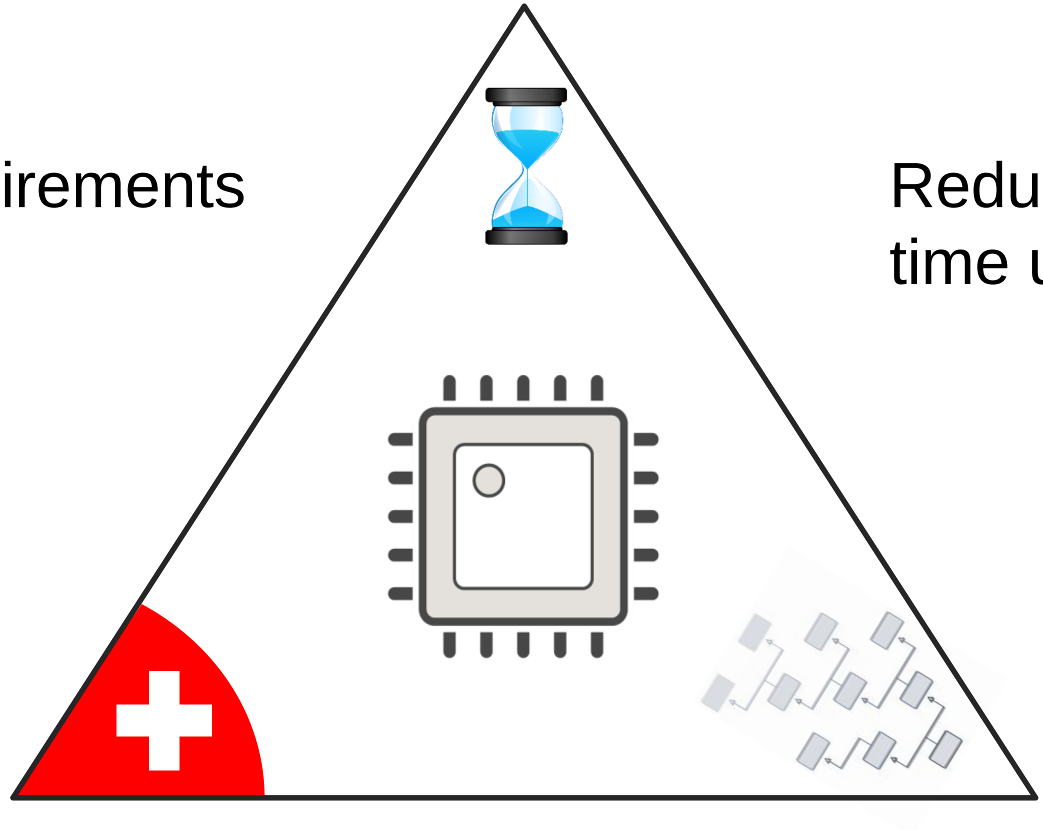
Vasco Lenzi



Key Takeaways

Meet stringent requirements and lower costs

Reduce hardware testing time up to 5x



Manage design complexity and improve team collaboration

Punch Powertrain develops complex SoC-based motor control

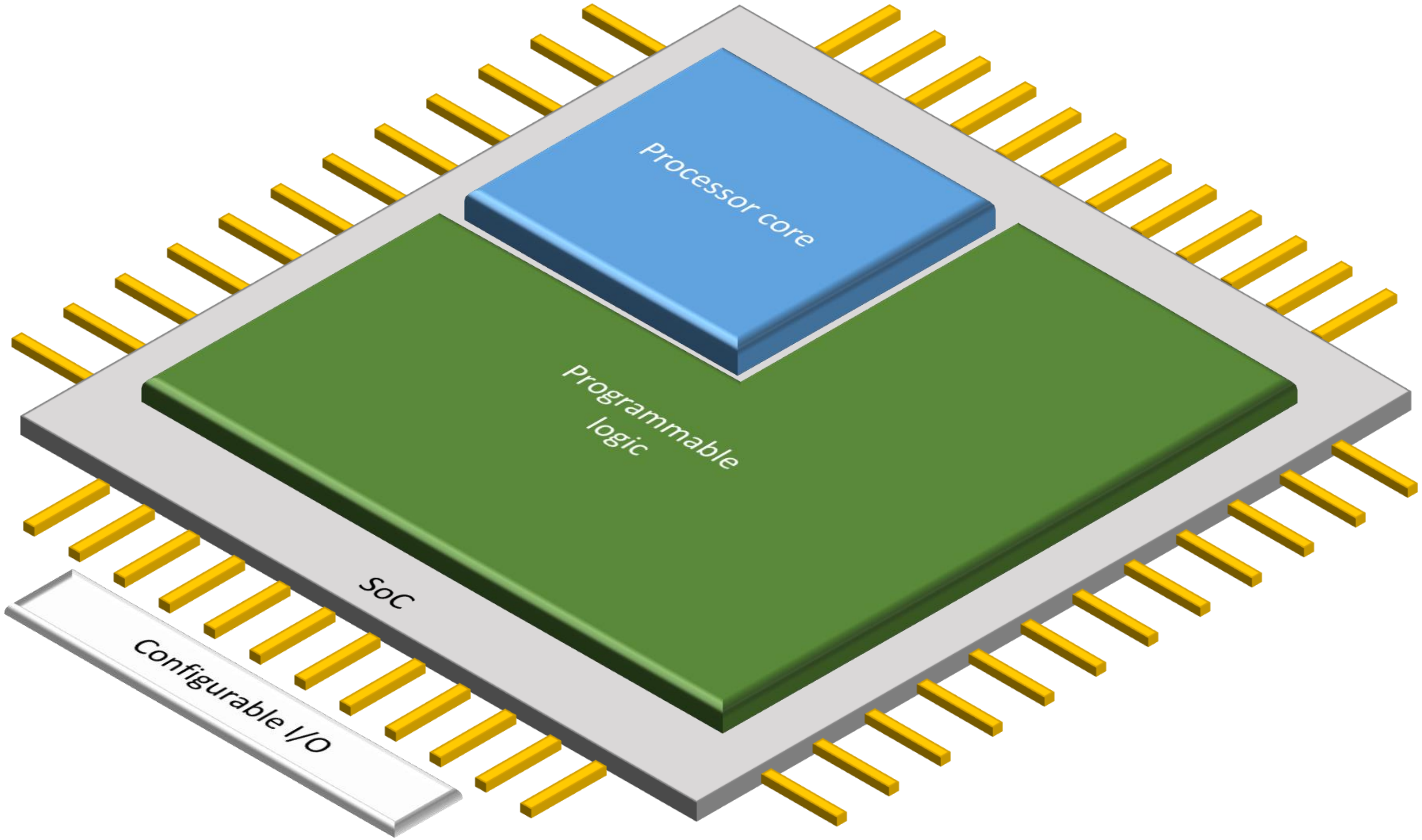
- Powertrains for hybrid and electric vehicles
- Hardware choice through simulations
- Traditional microcontroller too slow
- No experience designing FPGAs!

- ✓ 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

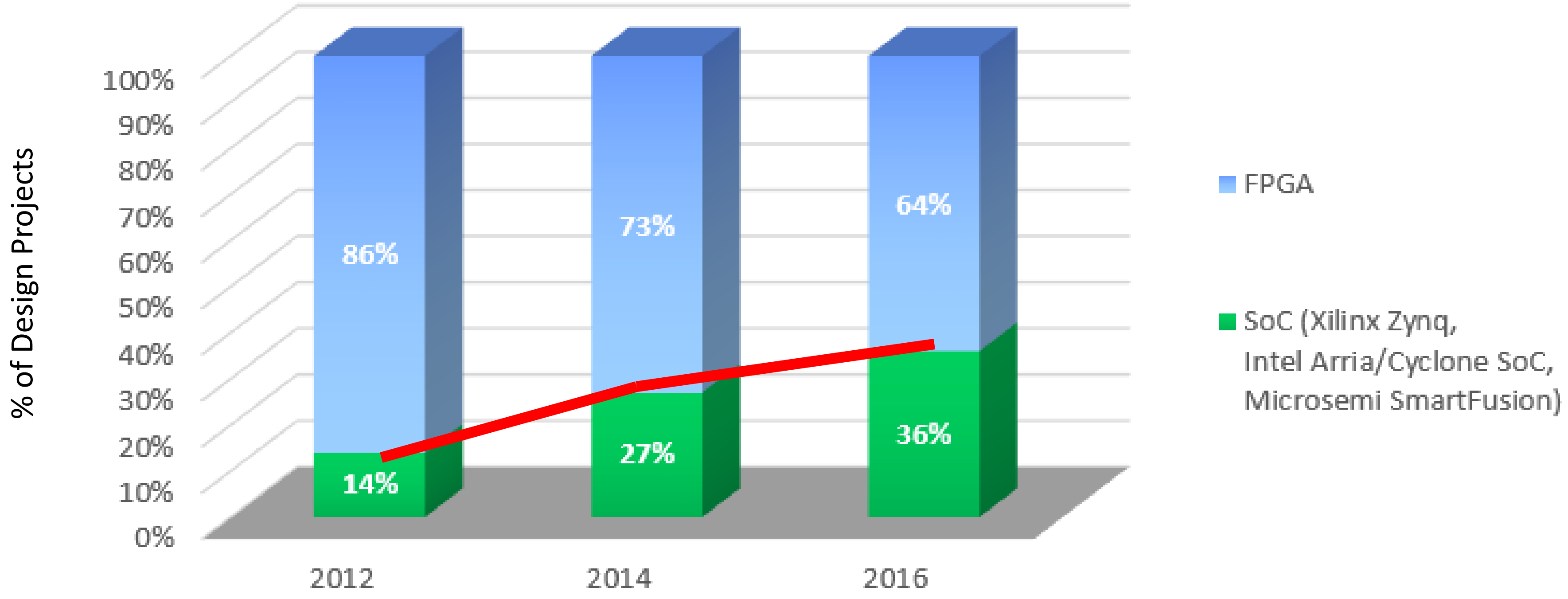
Link to video

Key trend: Increasing demands from motor drives

Systems-on-Chip for motor control

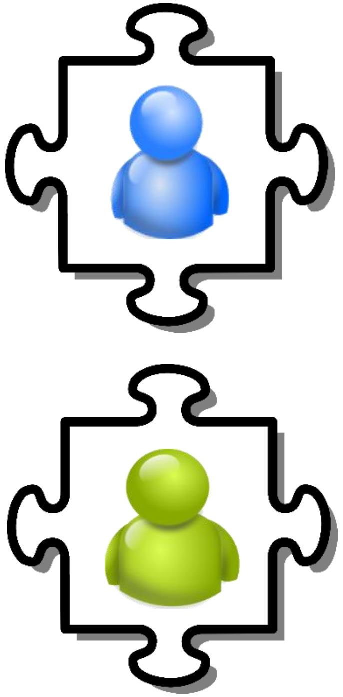


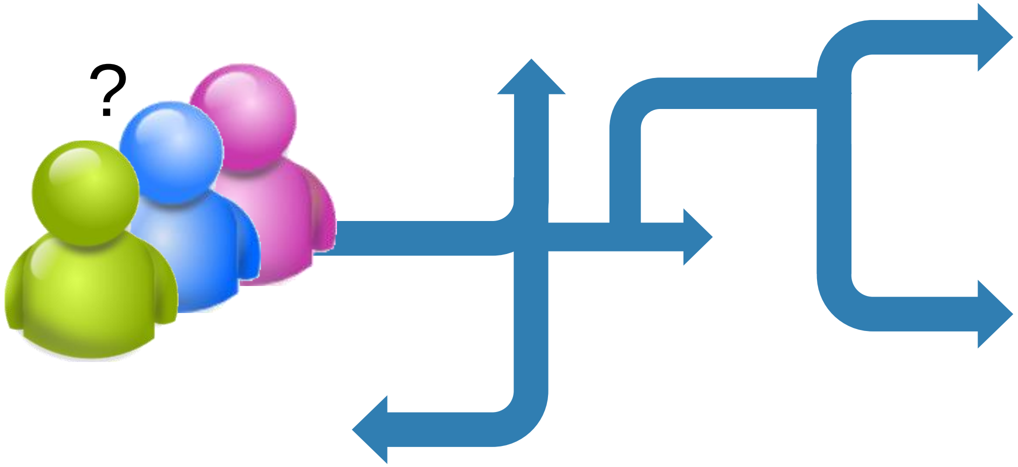
Key Trend: SoCs are now used in 36% of new FPGA projects



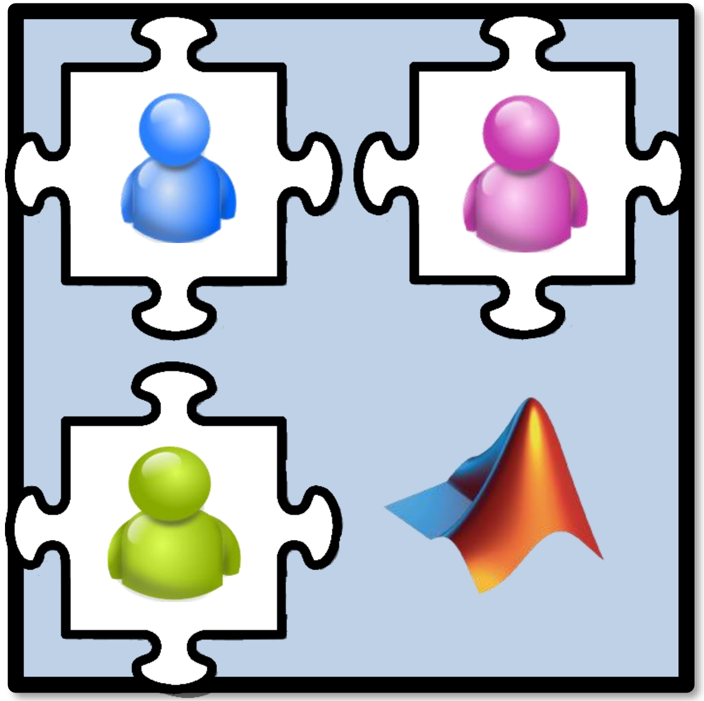
Source: Wilson Research Group and Mentor Graphics, 2016 Functional Verification Study

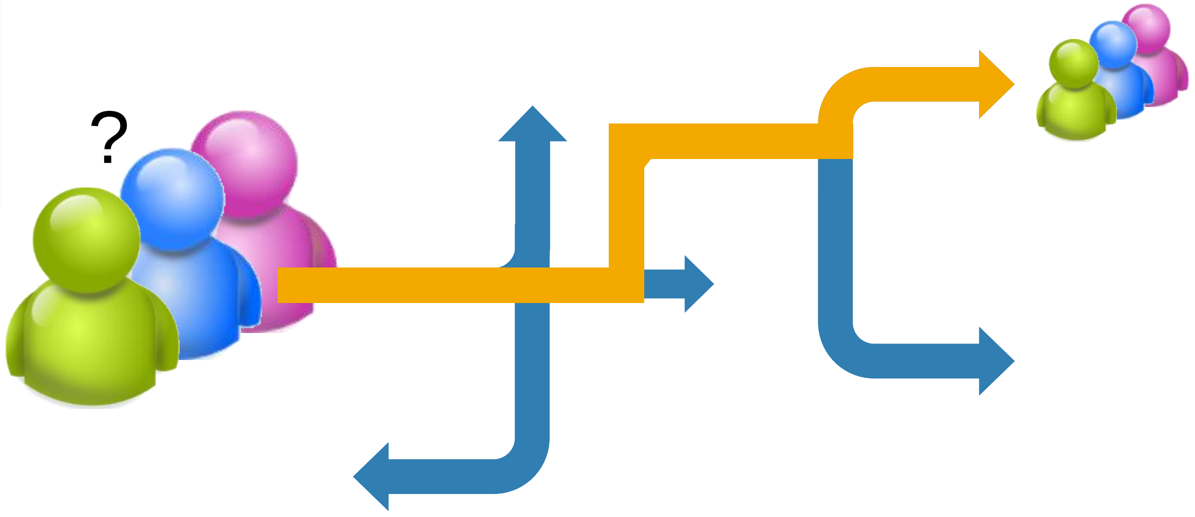
Challenges in using SoCs for Motor and Power Control

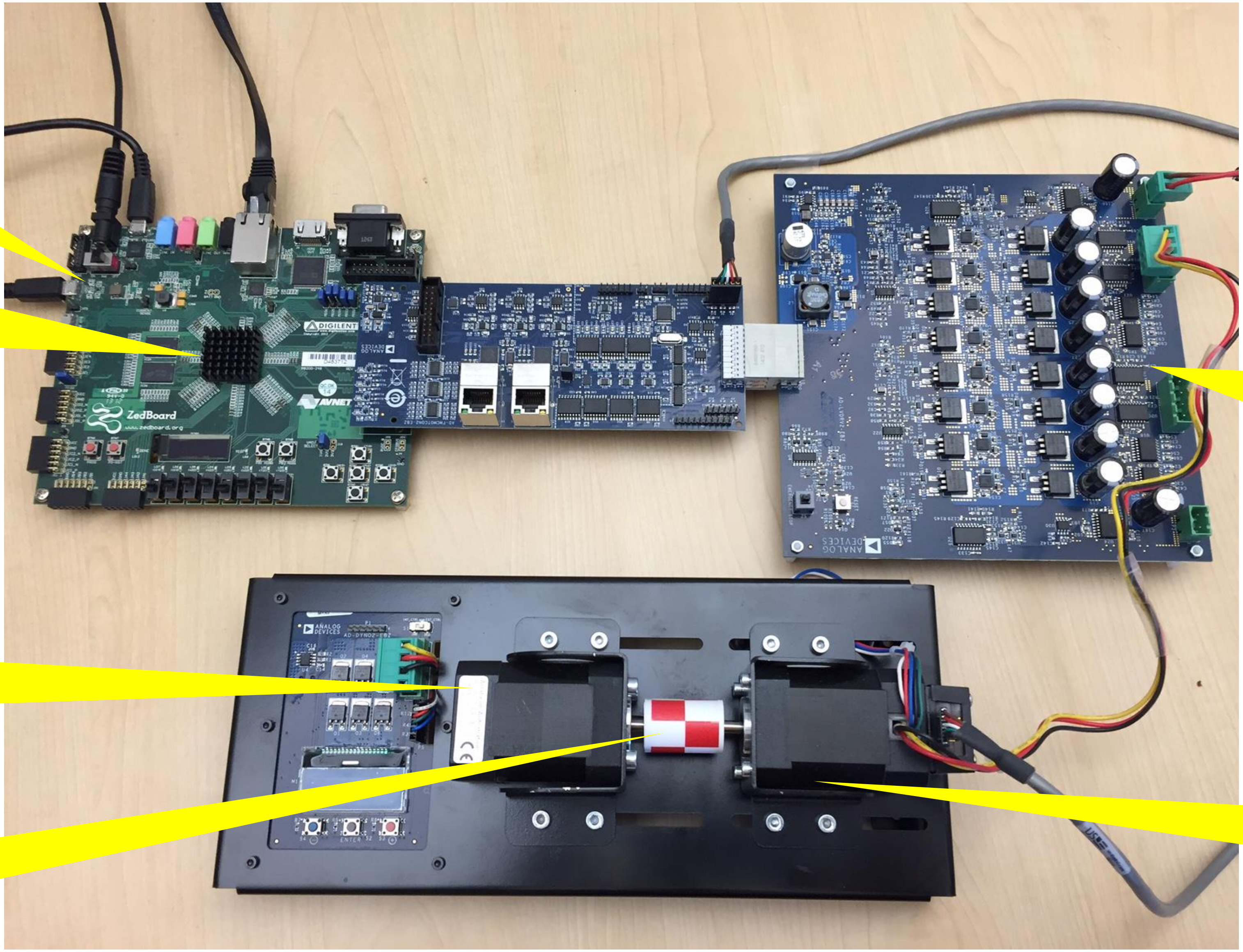




Why use Model-Based Design to develop motor control applications on SoCs?







ZedBoard

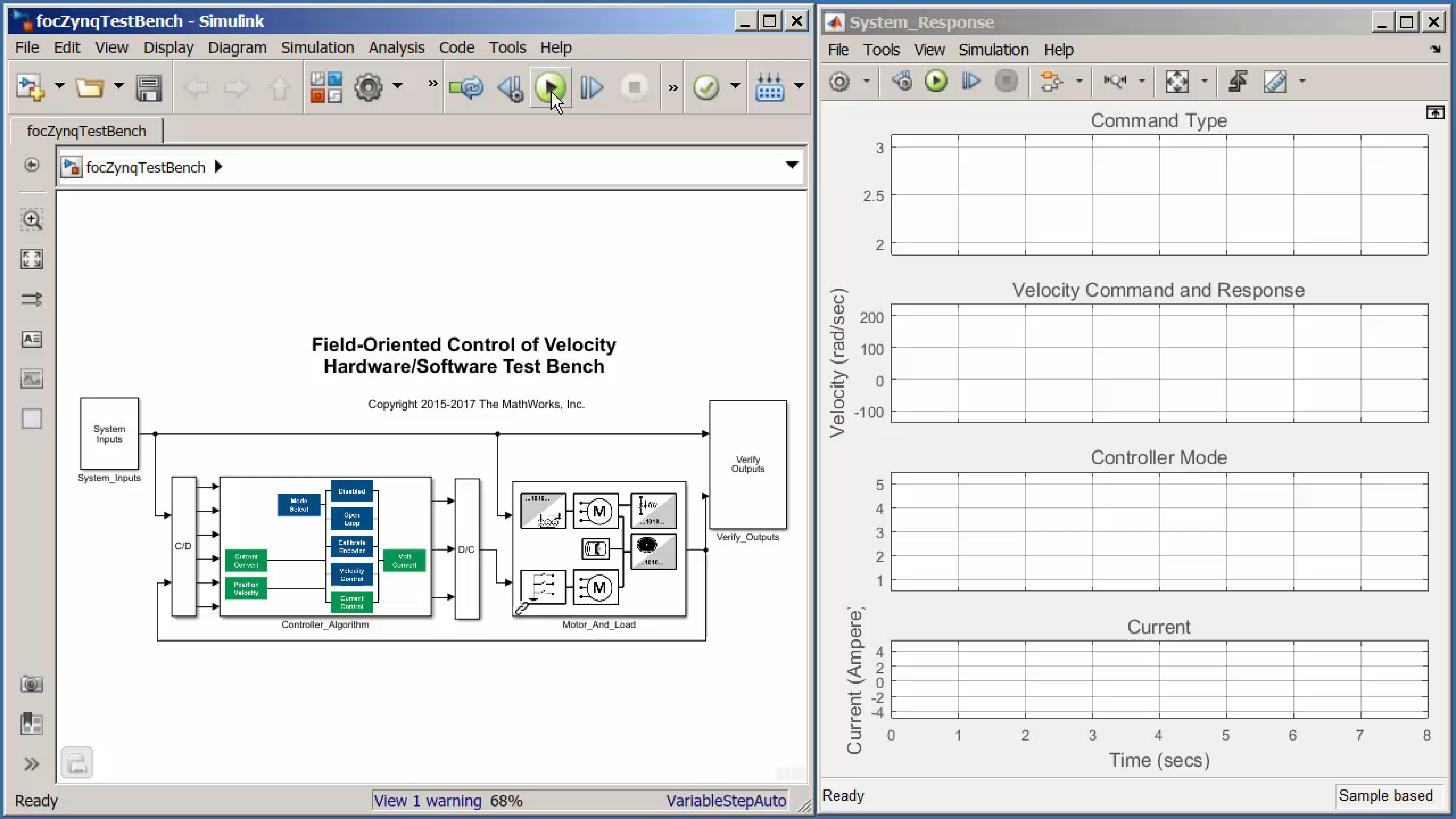
Zynq SoC
(XC7Z020)

Load motor

Mechanical coupler

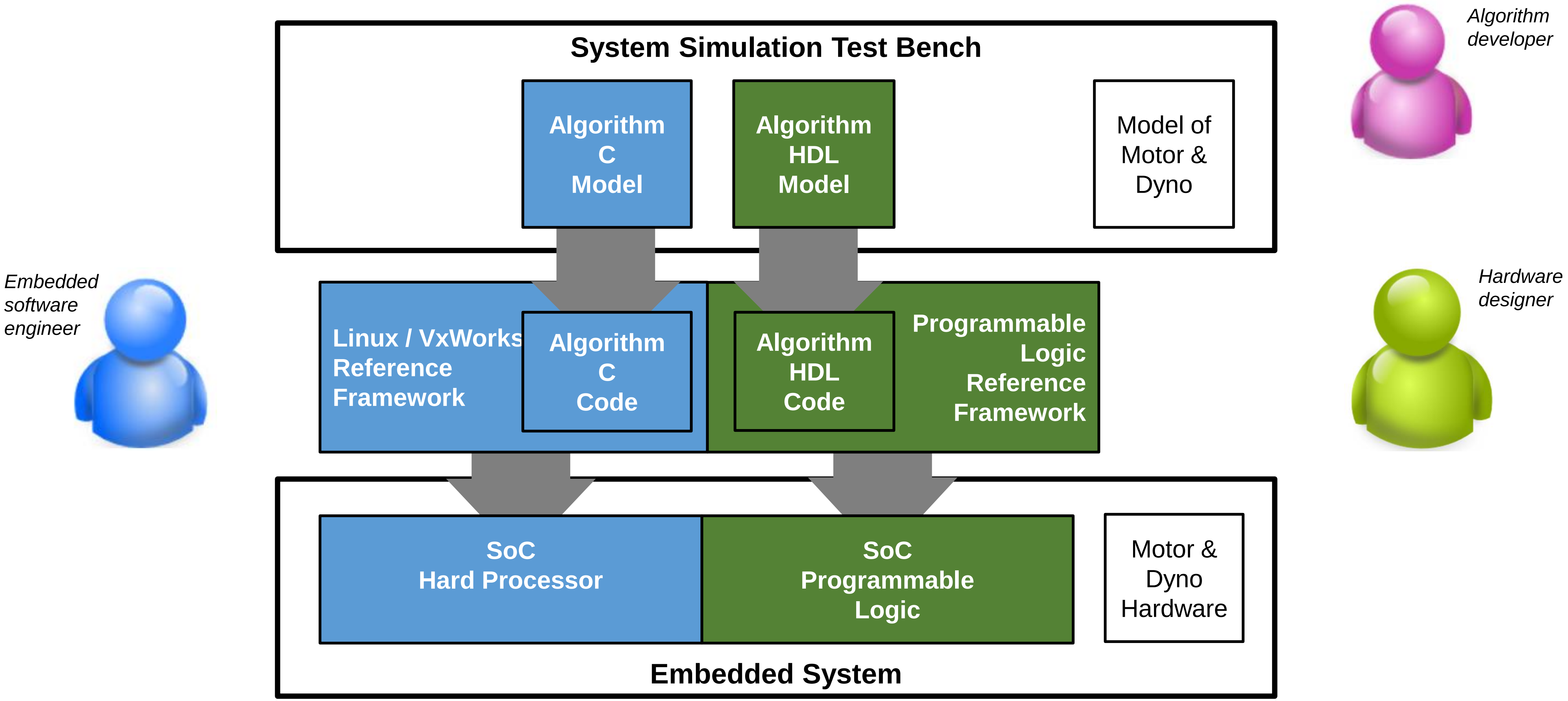
FMC module:
control board +
low-voltage board

Motor under test
(with encoder)

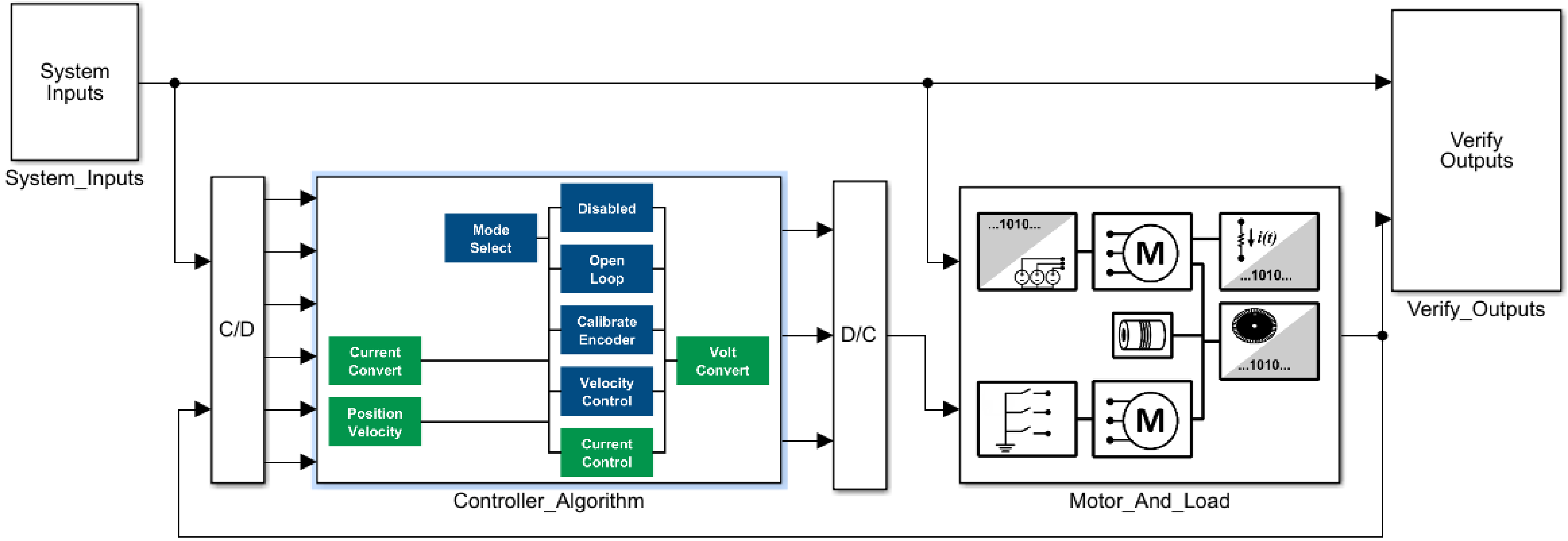


focZynqTestBench - Simulink
File Edit View Display Diagram Simulation Analysis Code Tools Help
focZynqTestBench
focZynqTestBench ▶
Field-Oriented Control of Velocity
Hardware/Software Test Bench
Copyright 2015-2017 The MathWorks, Inc.
System Inputs
System_Inputs
C/D
Mode Select
Disabled
Open Loop
Calibrate Encoder
Velocity Control
Current Control
Current Convert
Position Velocity
Volt Convert
Controller_Algorithm
D/C
M
M
Motor_And_Load
Verify Outputs
Verify_Outputs
Ready
View 1 warning 68%
VariableStepAuto
System_Response
File Tools View Simulation Help
Command Type
3
2.5
2
Velocity Command and Response
Velocity (rad/sec)
200
100
0
-100
Controller Mode
5
4
3
2
1
Current
Current (Ampere)
4
2
0
-2
-4
0
1
2
3
4
5
6
7
8
Time (secs)
Ready
Sample based

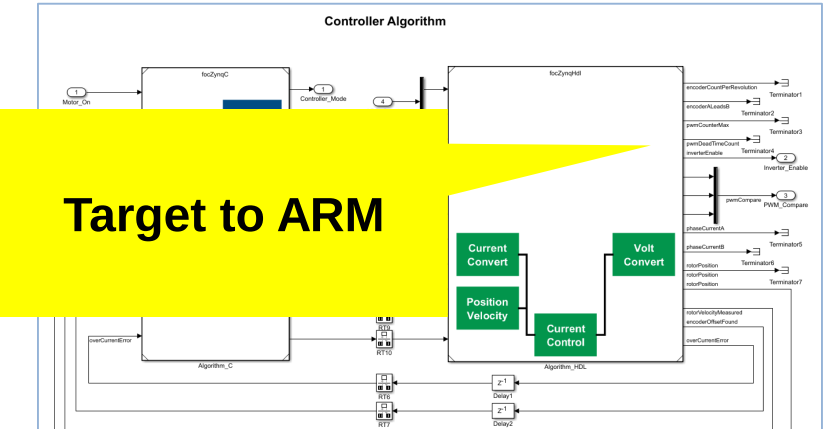
Conceptual workflow targeting SoCs



Hardware/software partitioning

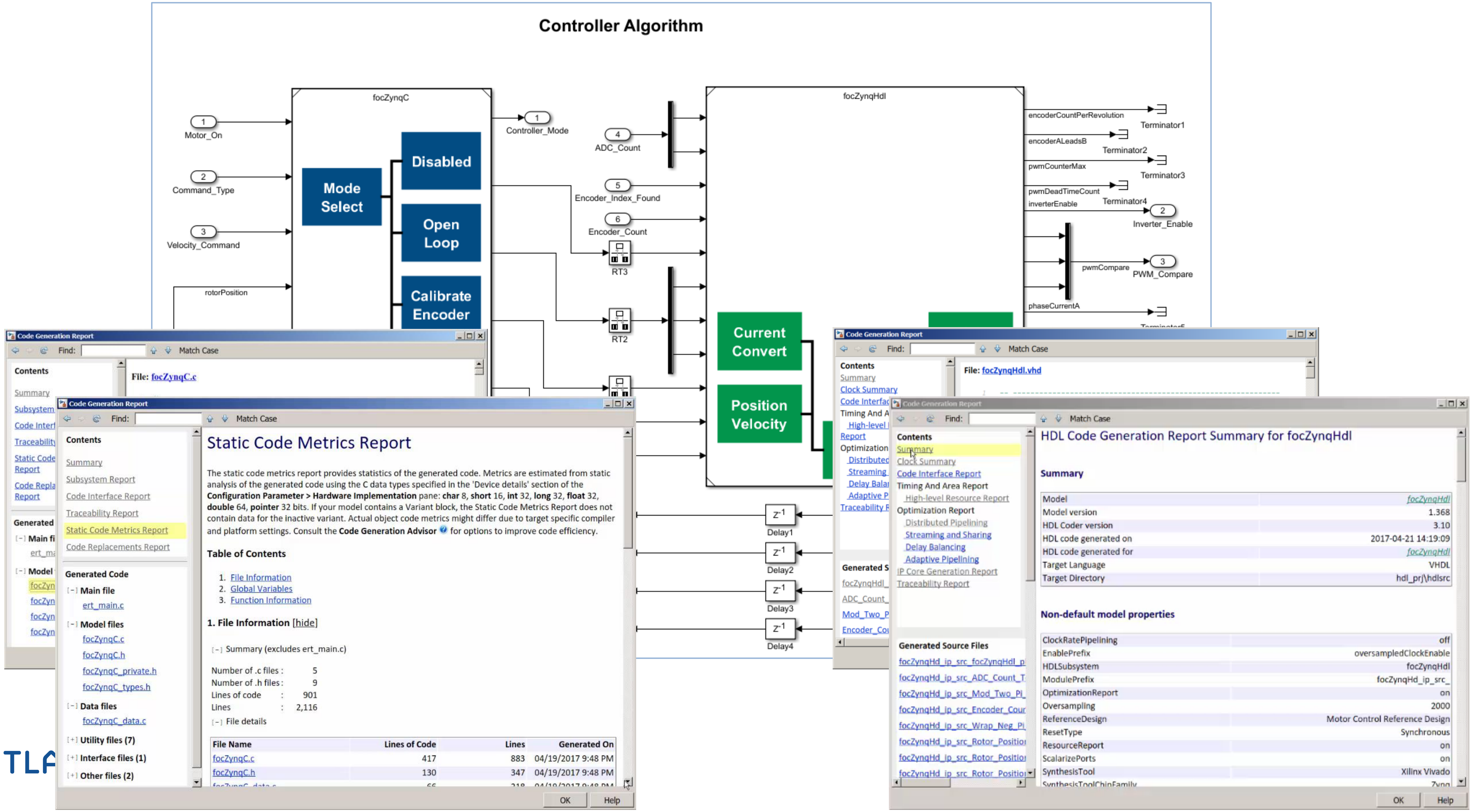


Target to Programmable Logic



Target to ARM

Code Generation



Code Generation Report

Contents

- Summary
- Subsystem Report
- Code Interface Report
- Traceability Report
- Static Code Metrics Report
- Code Replacements Report

Generated Code

[-] Main file

- ert_main.c

[-] Model files

- focZynqC.c
- focZynqC.h
- focZynqC_private.h
- focZynqC_types.h

[-] Data files

- focZynqC_data.c

[+] Utility files (7)

[+] Interface files (1)

[+] Other files (2)

Static Code Metrics Report

The static code metrics report provides statistics of the generated code. Metrics are estimated from static analysis of the generated code using the C data types specified in the 'Device details' section of the **Configuration Parameter > Hardware Implementation** pane: **char** 8, **short** 16, **int** 32, **long** 32, **float** 32, **double** 64, **pointer** 32 bits. If your model contains a Variant block, the Static Code Metrics Report does not contain data for the inactive variant. Actual object code metrics might differ due to target specific compiler and platform settings. Consult the **Code Generation Advisor** for options to improve code efficiency.

Table of Contents

1. File Information
2. Global Variables
3. Function Information

1. File Information [hide]

[-] Summary (excludes ert_main.c)

```
Number of .c files :      5
Number of .h files :      9
Lines of code      :    901
Lines              :  2,116
```

[-] File details

File Name	Lines of Code	Lines	Generated On
focZynqC.c	417	883	04/19/2017 9:48 PM
focZynqC.h	130	347	04/19/2017 9:48 PM

Code Generation Report

Contents

- Summary
- Clock Summary
- Code Interface Report
- Timing And Area Report
  - High-level Resource Report
- Optimization Report
  - Distributed Pipelining
  - Streaming and Sharing
  - Delay Balancing
  - Adaptive Pipelining
- IP Core Generation Report
- Traceability Report

Generated Source Files

- focZynqHd_ip_src_focZynqHdl_p
- focZynqHd_ip_src_ADC_Count_T
- focZynqHd_ip_src_Mod_Two_Pi_
- focZynqHd_ip_src_Encoder_Cour
- focZynqHd_ip_src_Wrap_Neg_Pi
- focZynqHd_ip_src_Rotor_Positio
- focZynqHd_ip_src_Rotor_Positio
- focZynqHd_ip_src_Rotor_Positio

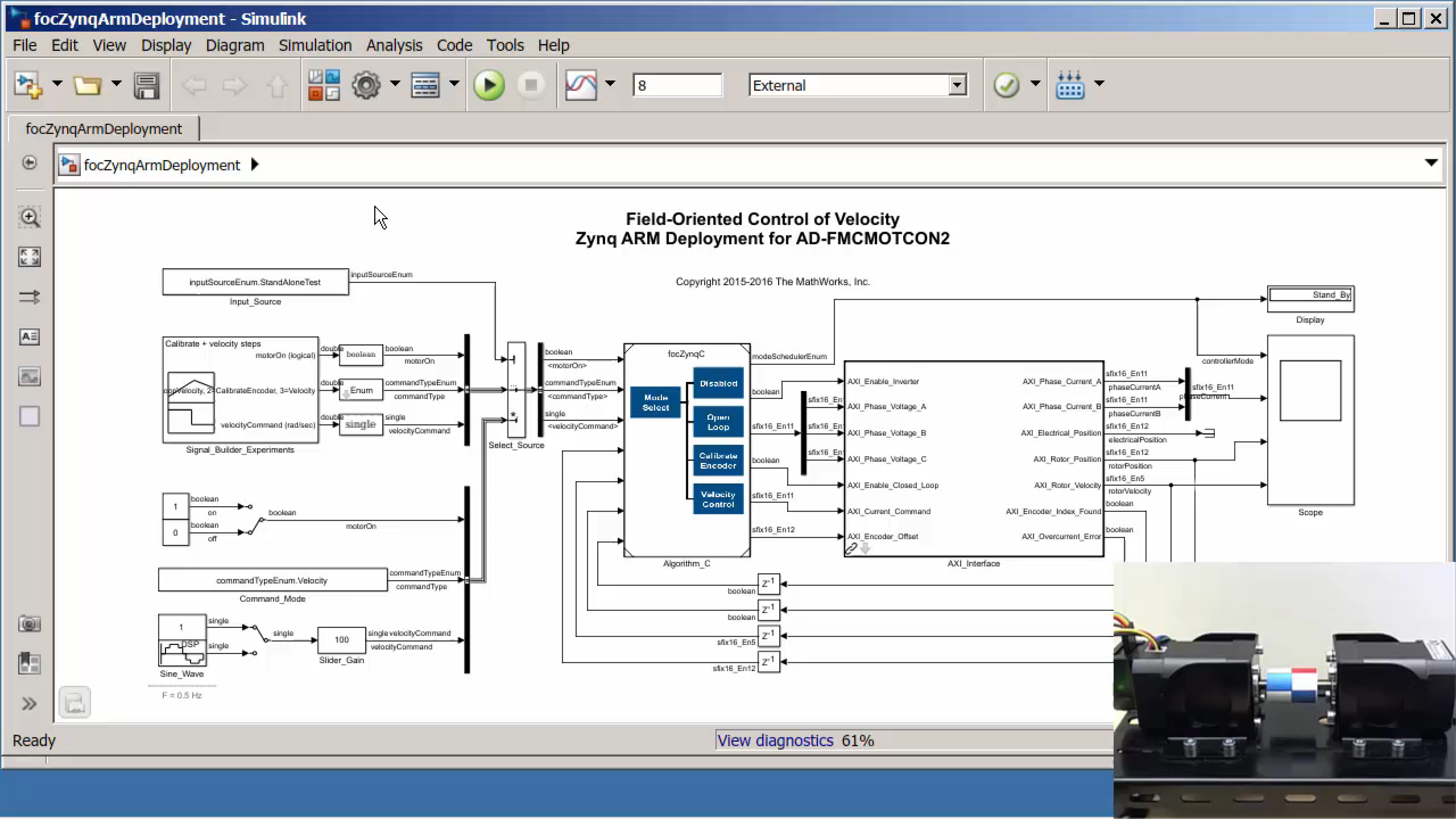
HDL Code Generation Report Summary for focZynqHdl

Summary

Model	*focZynqHdl*
Model version	1.368
HDL Coder version	3.10
HDL code generated on	2017-04-21 14:19:09
HDL code generated for	*focZynqHdl*
Target Language	VHDL
Target Directory	hdl_prj\hdlsrc

Non-default model properties

ClockRatePipelining	off
EnablePrefix	oversampledClockEnable
HDLSubsystem	focZynqHdl
ModulePrefix	focZynqHd_ip_src_
OptimizationReport	on
Oversampling	2000
ReferenceDesign	Motor Control Reference Design
ResetType	Synchronous
ResourceReport	on
ScalarizePorts	on
SynthesisTool	Xilinx Vivado
SynthesisToolChipFamily	Zynq



focZynqArmDeployment - Simulink

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

Copyright 2015-2016 The MathWorks, Inc.

Ready

View diagnostics 61%

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

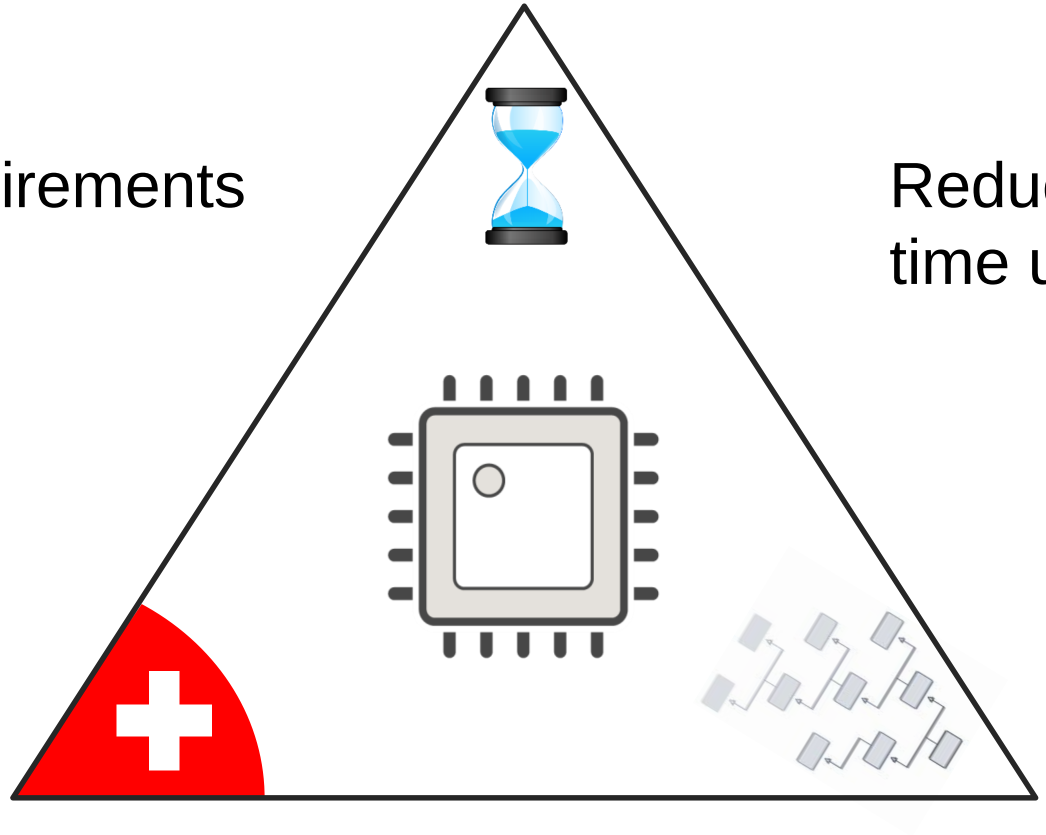
A SCARA robot.

> **“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 lower costs



Reduce hardware testing time up to 5x

Manage design complexity and improve team collaboration

Learn More

- Get an in-depth demo in the Technology Showcase
  - discuss the award-winning Native Floating Point in HDL Coder!
- Videos
  - HDL Coder: Native Floating Point
- Webinars
  - Prototyping SoC-based Motor Controllers on Intel SoCs with MATLAB and Simulink
  - How to Build Custom Motor Controllers for Zynq SoCs with MATLAB and Simulink
- Articles
  - How Modeling Helps Embedded Engineers Develop Applications for SoCs (MATLAB Digest)
  - MATLAB and Simulink Aid HW-SW Codesign of Zynq SoCs (Xcell Software Journal)
- Tutorials:
  - Define and Register Custom Board and Reference Design for SoC Workflow
  - Field-Oriented Control of a Permanent Magnet Synchronous Machine on SoCs