

# Model-Based Design: Generating Embedded Code for Prototyping or Production

**Ruth-Anne Marchant**  
**Application Engineer**  
**MathWorks**



## ABB Accelerates Application Control Software Development for Power Electronic Controller

### Challenge

Adopt a more efficient development process using tools that accelerate the design of new application software for a high-powered electronic controller for power converters

### Solution

Use MathWorks tools to design and validate their control algorithms while streamlining the application software development process for the controller

### Results

- Development times and costs reduced
- Development process improved
- Highly accurate code generated



AC 800PEC controller.

**“Our system engineers can program, simulate, and verify the AC 800PEC controller’s regulation software very rapidly in MATLAB and Simulink.”**

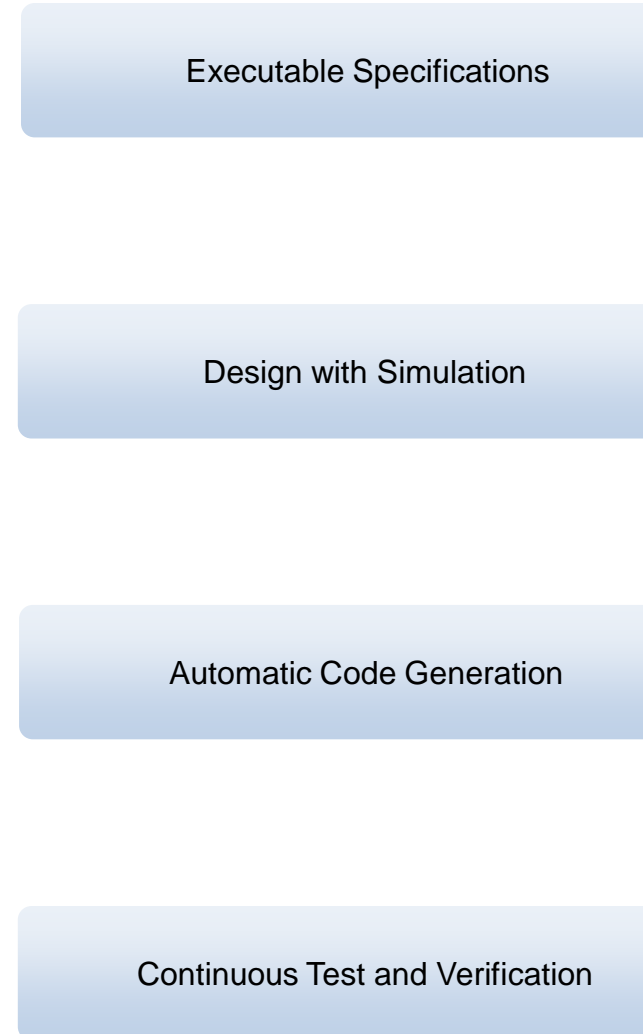
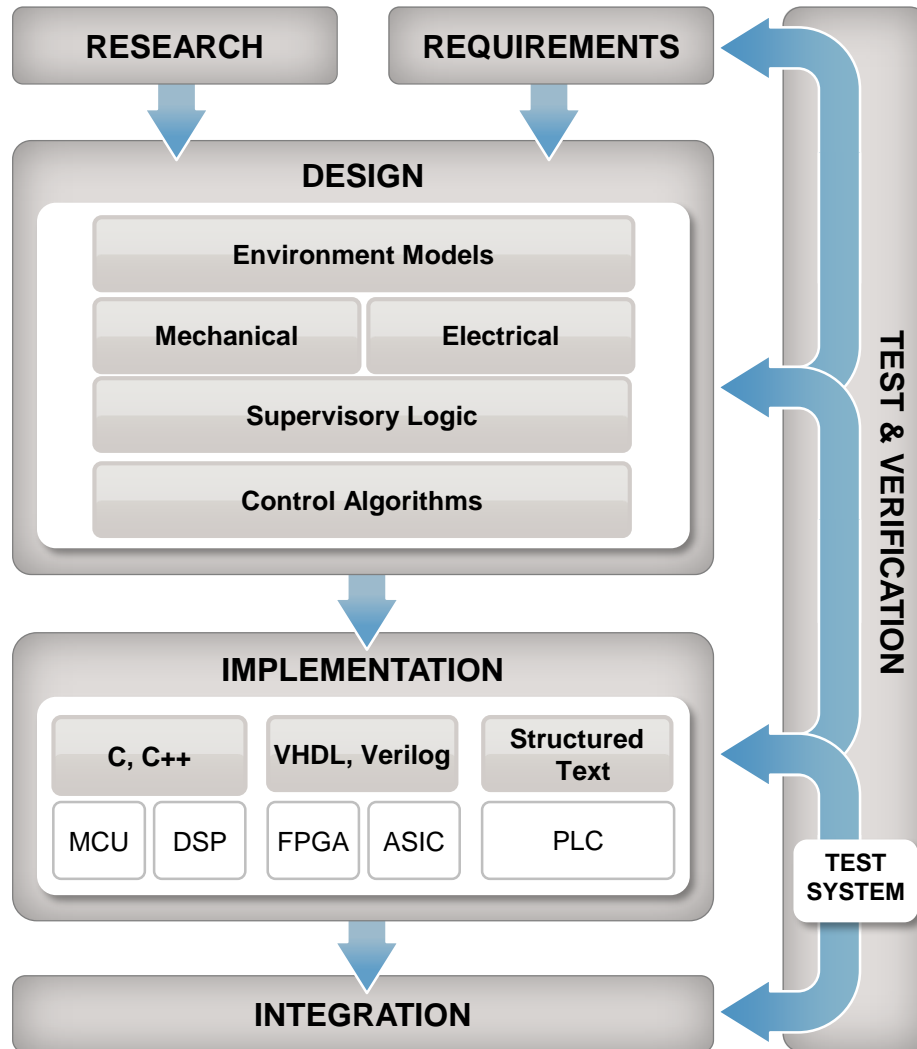
**Fritz Wittwer  
ABB**

[Link to user story](#)

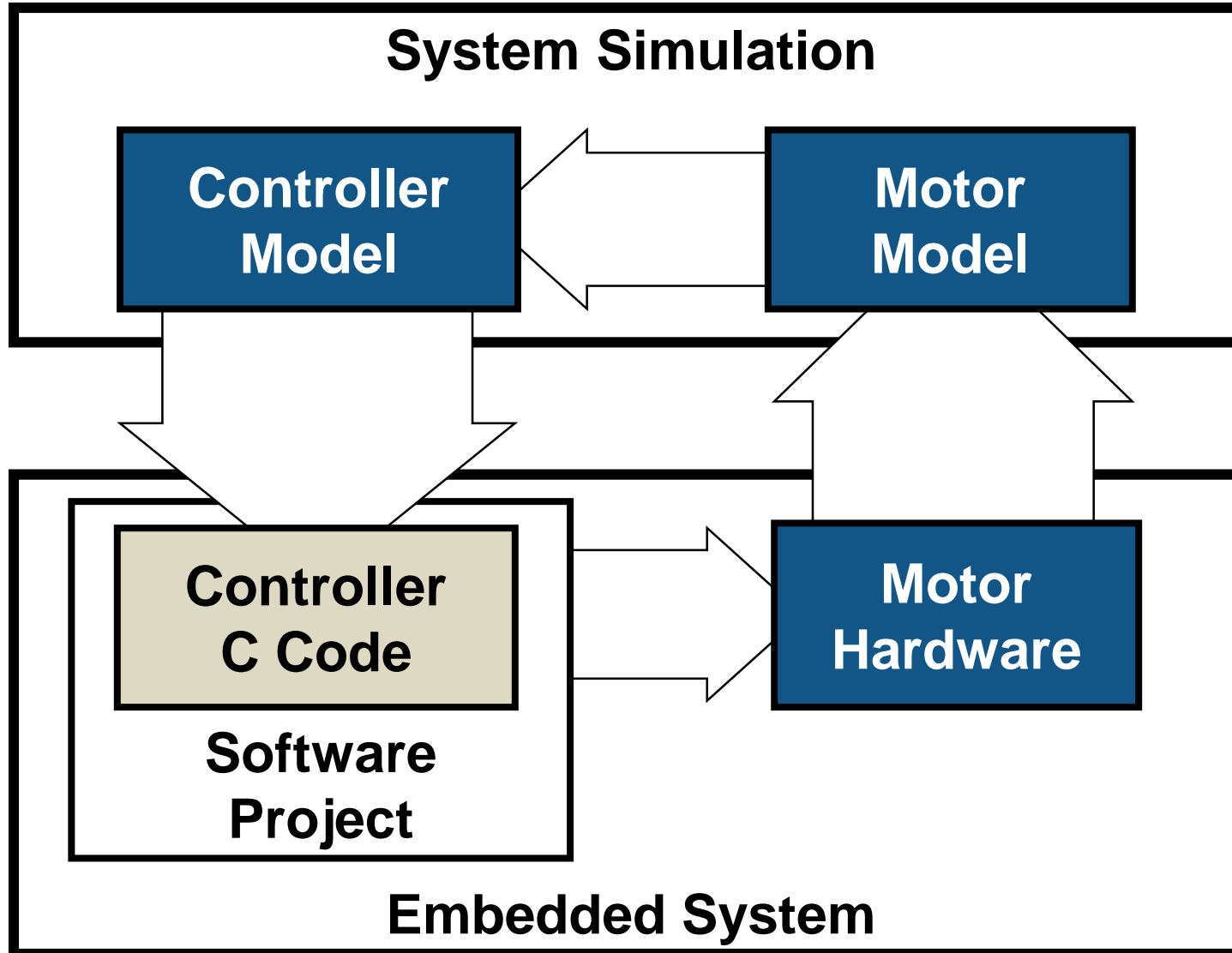
# Outline

- Recap of Model-Based Design
- Generating code for rapid prototyping
- Generating code for production software
  - Preparing Model for Embedded Code Generation
  - Evolving Model for Fixed Point Implementation
- Summary

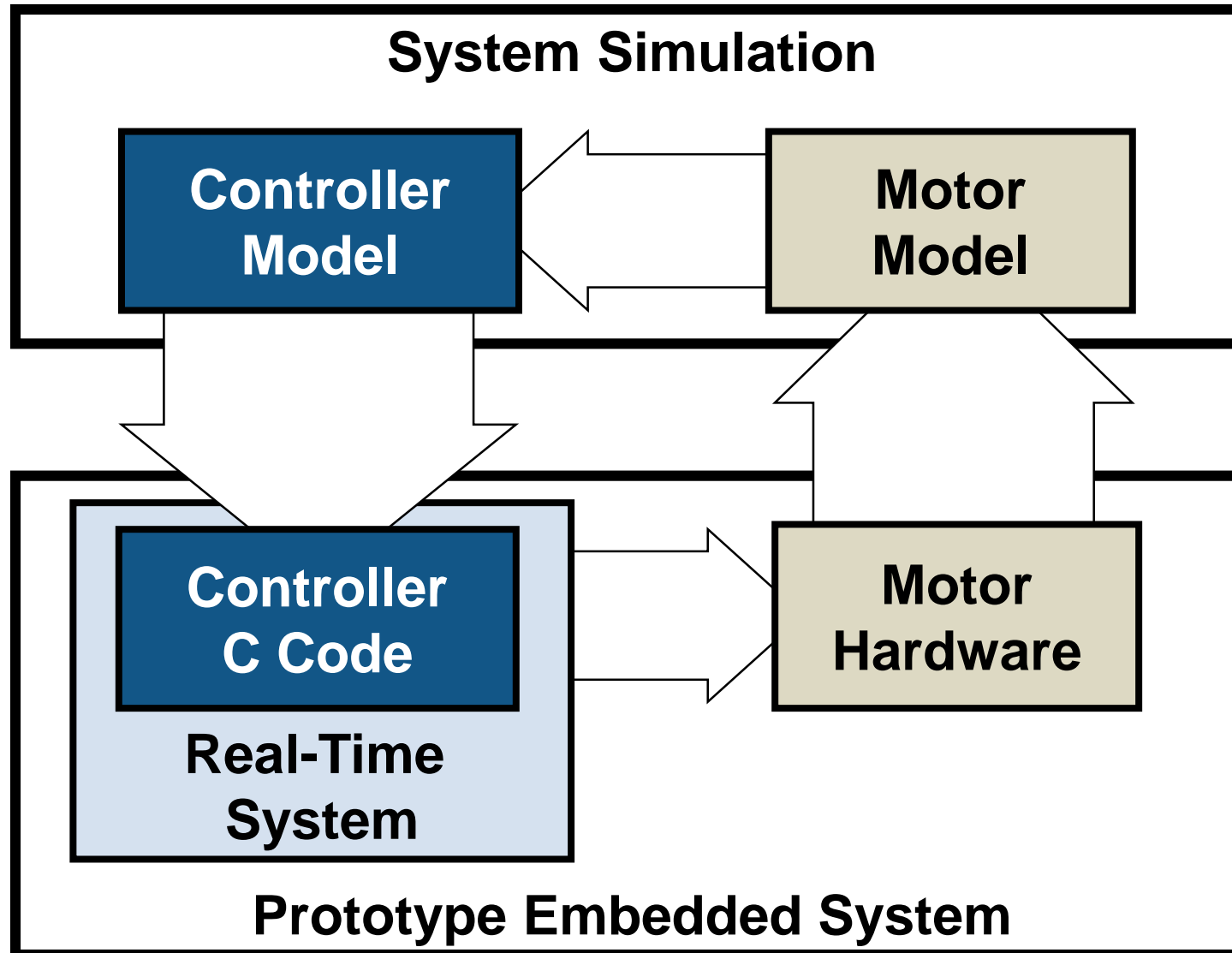
# Adopting Model-Based Design



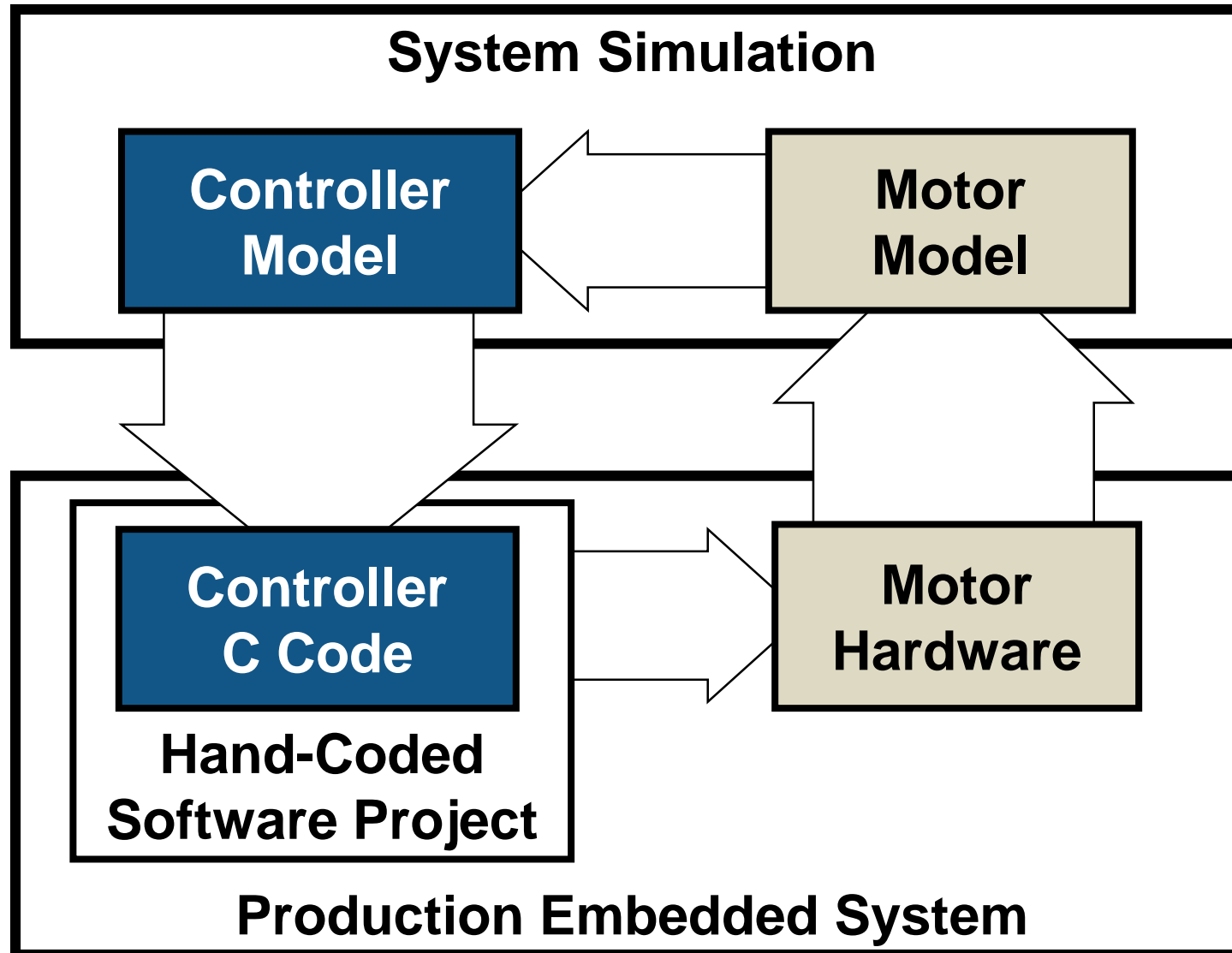
# Motor Control System Design



# Automatic Code Generation: Prototype on real-time hardware



# Automatic Code Generation: Generate code for production



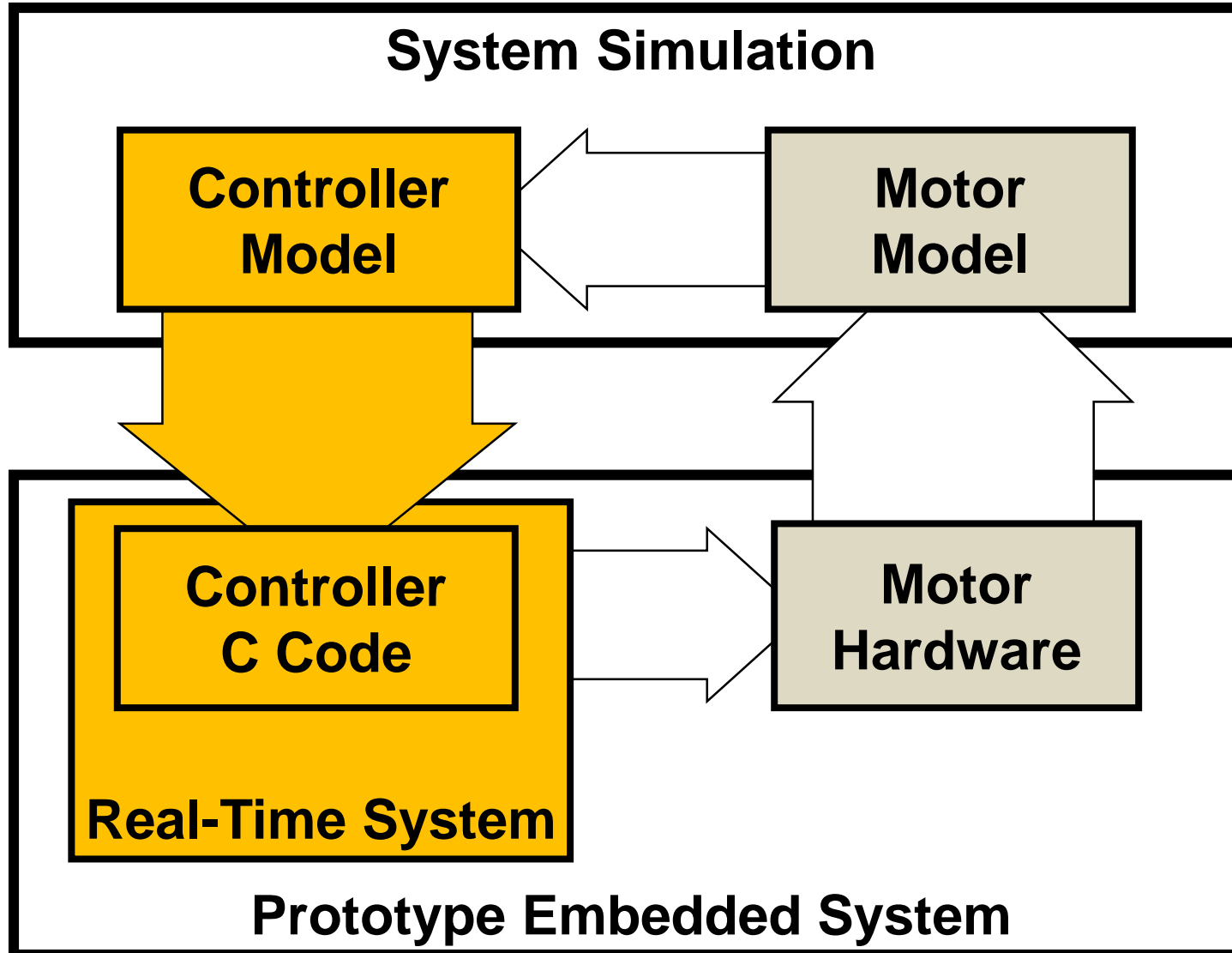


# Outline

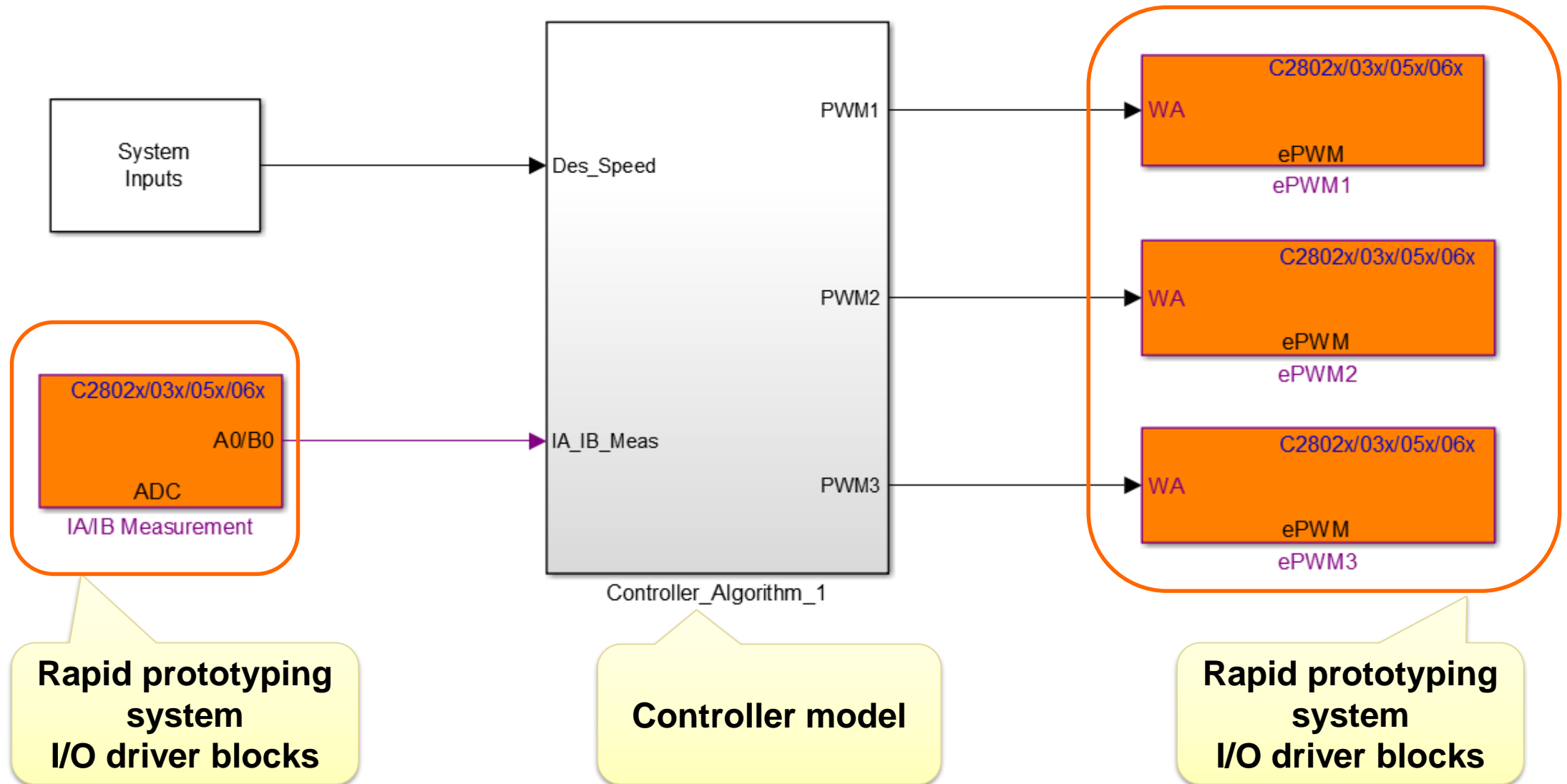
- Recap of Model-Based Design
- **Generating code for rapid prototyping**
- Generating code for production software
  - Preparing Model for Embedded Code Generation
  - Evolving Model for Fixed Point Implementation
- Summary

# Automatic Code Generation with Simulink

## Prototype on real-time hardware



# Rapid Prototyping Model for Texas Instrument C2000 F28069M LaunchPad™ Development Kit

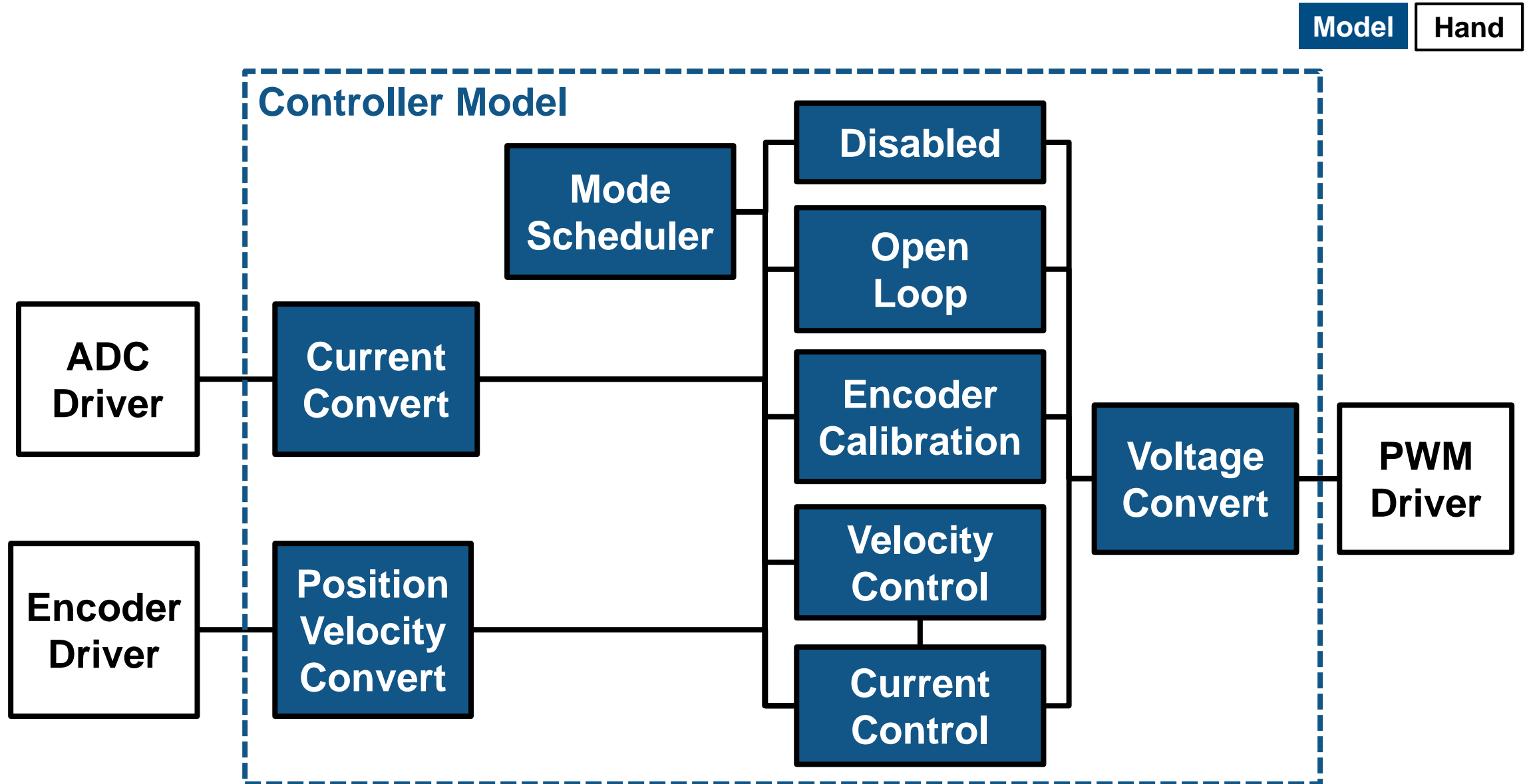


# **DEMO: Motor control using Embedded Coder Support Package for Texas Instruments C2000 Processors**

# Outline

- Recap of Model-Based Design
- Generating code for rapid prototyping
- **Generating code for production software**
  - Preparing Model for Embedded Code Generation
  - Evolving Model for Fixed Point Implementation
- Summary

# Controller Model for Production Code Generation



# Integrate generated controller code with your hand-coded software project

Model

Hand

## Embedded Software Project Pseudo-Code

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

    controllerInit();

    while(1) {
    }
}
```

```
interruptServiceRoutine()
{
    readAdcCountFromDriver();
    readEncoderCountFromDriver();

    controller();

    writePwmCountToDriver();
}
```

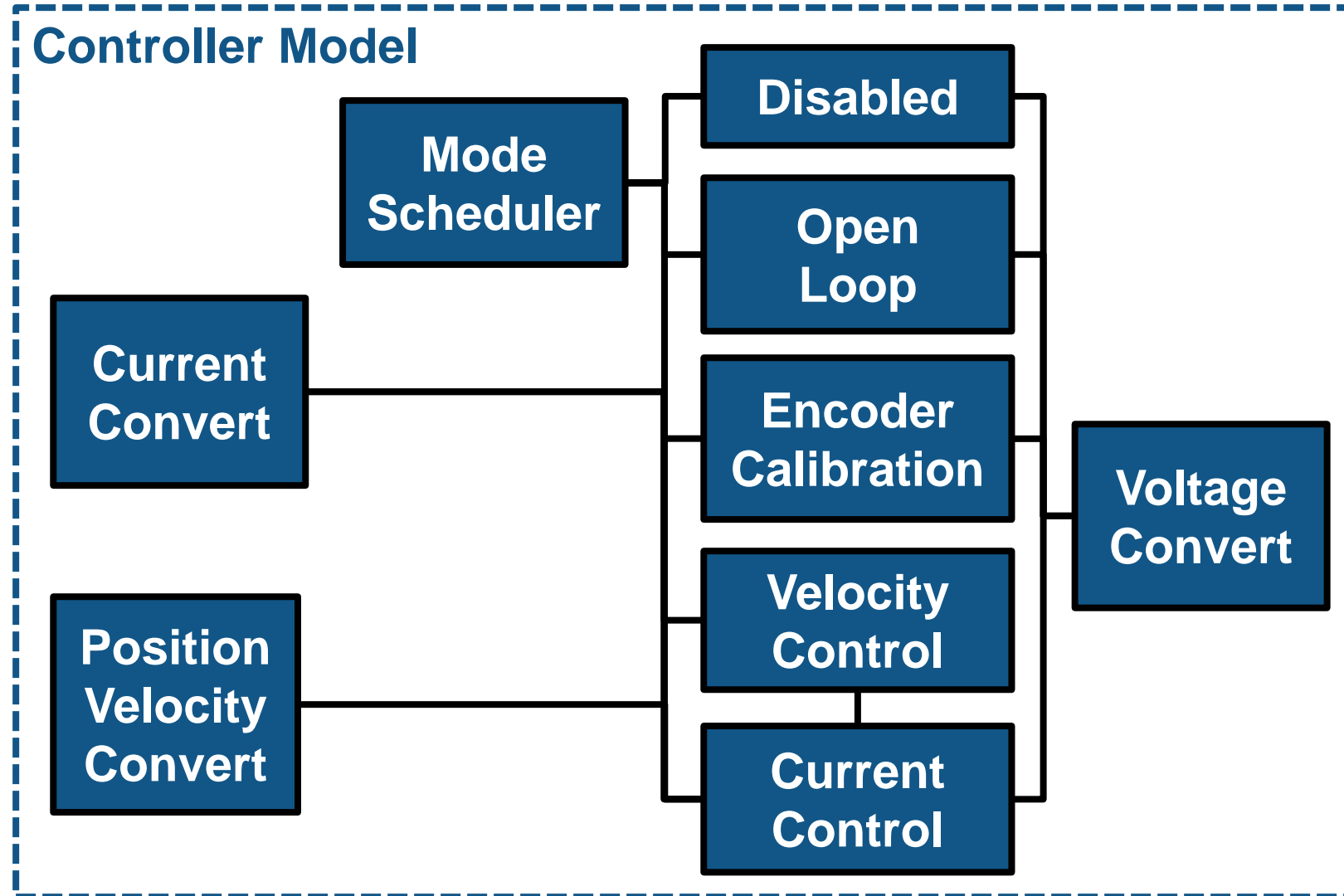
# **DEMO: Prepare algorithm model to generate embedded code and specify code interface**



# Outline

- Recap of Model-Based Design
- Generating code for rapid prototyping
- Generating code for production software
  - Preparing Model for Embedded Code Generation
  - Evolving Model for Fixed Point Implementation
- Summary

# Design for fixed-point implementation



# Design for fixed-point implementation at component level

**Controller Model**

**Velocity  
Control**

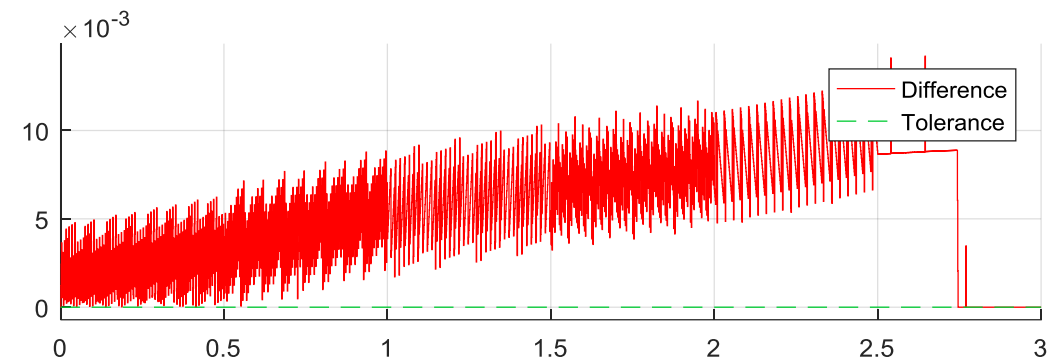
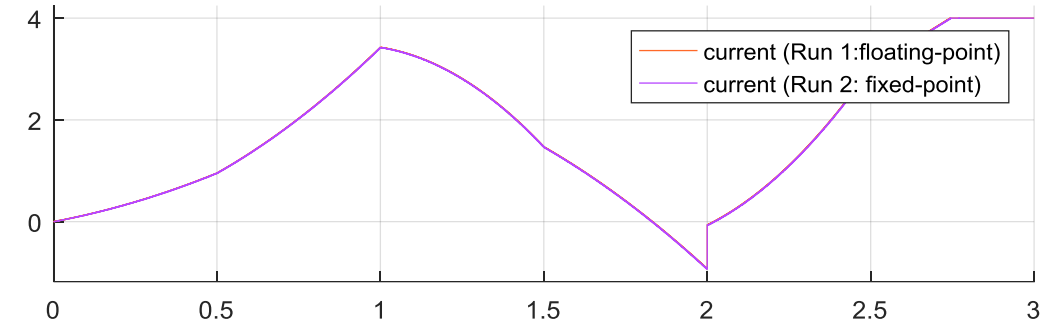
# Design for Fixed-Point Implementation - Workflow

- Set up model to use Fixed-Point Tool
  - Specify minimum and maximum values on model inputs
- Using Fixed-Point Tool:
  - Select system under design
  - Derive minimum and maximum values
  - Propose data types
    - Determined based on range data
  - Apply proposed data types

The screenshot displays the 'Workflow' tab of the Fixed-Point Tool interface. The 'System under design' section is highlighted with a green bar and an orange box, with an arrow pointing to it. Below this, the 'Fixed-point preparation' section includes a 'Fixed-Point Advisor' icon. The 'Configure model settings' section has three options: 'Range collection using double override', 'Range collection with specified data types', and 'Remove overrides and disable range collection'. The 'Range collection' section shows 'Run name: Run 1' and a 'Simulate model' button. Below that, 'Derive ranges for: System Under Design' is highlighted with an orange box and an arrow. The 'Automatic data typing' section includes checkboxes for 'Signedness', 'Word length', and 'Fraction length', and 'Propose for: Inherited' and 'Floating point'. It also has a 'Default word length' field set to 16, a 'When proposing types use:' dropdown set to 'All collected ranges', and a 'Safety margin for simulation min/max (%)' field set to 0. At the bottom, two buttons are highlighted with orange boxes and arrows: 'Propose data types' and 'Apply accepted data types'.

# Design for Fixed-Point Implementation - Workflow

- Compare against baseline floating point design
  - Simulate fixed-point design and compare against floating-point design
- Explore trade-offs in design decisions
  - Test 16 vs 32 bit fixed point designs
- Integrate component design into system-level simulation to validate design decisions



# Outline

- Recap of Model-Based Design
- Generating code for rapid prototyping
- Generating code for production software
  - Preparing Model for Embedded Code Generation
  - Evolving Model for Fixed Point Implementation
- Summary

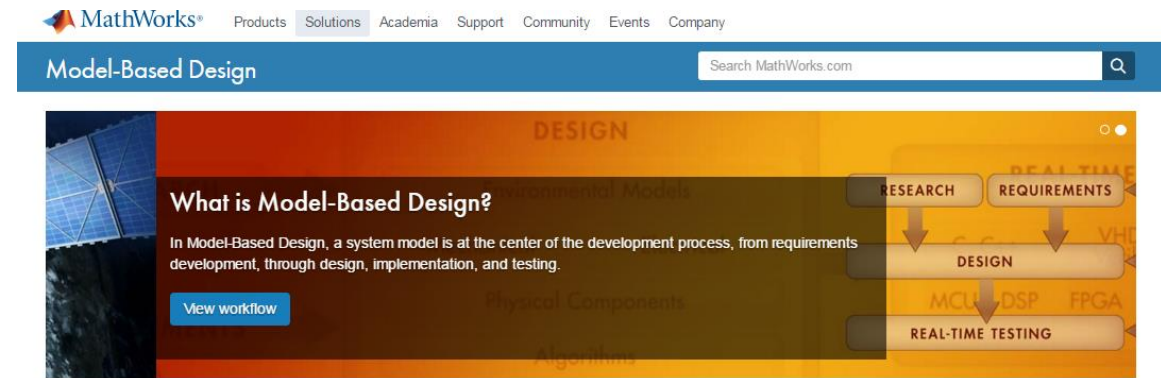
# Key Points

- Simulink is a multi-domain modelling and simulation environment that supports Model-Based Design
  
- Code generation technology can be used to
  - Quickly perform design iterations and deploy to prototyping hardware
  - Eliminate hand-coding errors in production code
  - Remove barriers to communication between teams

# Call to Action

## Learn more about Model-Based Design with Simulink

- Explore our [website](http://au.mathworks.com)
  - [au.mathworks.com](http://au.mathworks.com)
  
- Contact me:
  - Ruth-Anne Marchant
    - [ruth-anne.marchant@mathworks.com.au](mailto:ruth-anne.marchant@mathworks.com.au)



### Why Use Model-Based Design?

Model-Based Design is transforming the way engineers and scientists work by moving design tasks from the lab and field to the desktop.

When software and hardware implementation requirements are included, such as fixed-point and timing behavior, you can **automatically generate code** for embedded deployment and create test benches for **system verification**, saving time and avoiding the introduction of manually coded errors.

Use Model-Based Design with **MATLAB®** and **Simulink®** to improve product quality and reduce development time by 50% or more.



Model-Based Design of Safety-Critical Avionics Systems (Highlights)



Weinmann Develops Life-Saving Transport Ventilator



Alstom Grid Develops HVDC Transmission Control System

With Model-Based Design, you can:

- Use a common design environment
- Link designs directly to requirements
- Integrate testing with design
- Refine algorithms through multidomain simulation
- Automatically generate embedded software code and documentation
- Develop and reuse test suites

[Explore Model-Based Design with Simulink](#)

[Contact sales](#)

[Request a trial](#)



# Q & A