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

“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

- RESEARCH
- REQUIREMENTS
- DESIGN
  - Environment Models
    - Mechanical
    - Electrical
  - Supervisory Logic
  - Control Algorithms
- IMPLEMENTATION
  - C, C++
  - VHDL, Verilog
  - Structured Text
  - MCU, DSP, FPGA, ASIC, PLC
- TEST & VERIFICATION
- TEST SYSTEM

Executable Specifications
Design with Simulation
Automatic Code Generation
Continuous Test and Verification
Motor Control System Design

System Simulation

Controller Model

Motor Model

Controller C Code

Motor Hardware

Software Project

Embedded System
Automatic Code Generation: Prototype on real-time hardware

System Simulation

Controller Model

Motor Model

Controller C Code

Real-Time System

Prototype Embedded System

Motor Hardware
Automatic Code Generation: Generate code for production

System Simulation

Controller Model

Motor Model

Controller C Code

Motor Hardware

Hand-Coded Software Project

Production Embedded System
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

Prototype Embedded System

System Simulation

Controller Model

Motor Model

Controller C Code

Real-Time System

Motor 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

Controller Model

ADC Driver

Current Convert

Mode Scheduler

Disabled

Open Loop

Encoder Calibration

Voltage Convert

PWM Driver

Encoder Driver

Position Velocity Convert

Velocity Control

Current Control

Encoder Calibration

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

### Embedded Software Project Pseudo-Code

```c
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

Controller Model

- Mode Scheduler
  - Disabled
    - Open Loop
      - Encoder Calibration
        - Velocity Control
          - Current Control
  - Current Convert
  - Position Velocity Convert
- Voltage Convert
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
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 – au.mathworks.com

- Contact me:
  - Ruth-Anne Marchant
    - ruth-anne.marchant@mathworks.com.au
Q & A