Model-Based Design For Motor Control Development

Jens Sorensen – Analog Devices
Noam Levine – MathWorks
Topics

- Developing motor control applications using Model-Based Design and automatic code generation
  - Hardware platform
  - Model-Based Design overview
  - Modeling of motor control systems
  - Simulation and automatic code generation
  - Running generic C-code on embedded platform
  - When to use MBD and when to use “traditional” design methods
ADI Demo Platforms

- Motor Drive hardware
- 3 phase PM motor
- PC with MATLAB®, Simulink®, and IAR Embedded Workbench
HW Platform

- 100V-250V AC input with PFC, 5A
- Isolated current, position, voltage and diagnostic feedback
- 3-phase, 0.55kW, permanent magnet synchronous motor
**Key Points**
- ARM CORTEX M4F
- 240MHz
- Two 16-bit ADCs
  - 380ns conversion time
- SINC filter
- 384kB SRAM
- 2MB Flash
- 20 DMA channels
- Peripherals for motor control
- Communication interfaces
MODEL-BASED DESIGN
Why Model-Based Design?

- Requirements Development
- System Simulation
- Automatic Code Generation
- Continuous Verification
The Traditional Design Process

- Requirements Phase
- Design Phase
- Realization Phase
- Testing Phase
Traditional Design of a Multi-Domain System

Research & Requirements

Hardware
- Requirements
- Design
- Realization
- Testing

Software
- Requirements
- Design
- Realization
- Testing

Mechanical
- Requirements
- Design
- Realization
- Testing

Integration, Test & Certification
The Cost of a Traditional Design Process

Relative Cost to Fix Defects per Phase Found

Source: ROI for IV&V, NASA
Model-Based Design

- Model multi-domain systems
- Explore and optimize system behavior in floating point and fixed point
- Collaborate across teams and continents

- Generate efficient code
- Explore and optimize implementation tradeoffs

- Automate regression testing
- Detect design errors
- Parameter tuning
- Support certification and standards
Model-Based Design for Embedded System Development

**Executable models**
- Unambiguous
- “One Truth”

**Simulation**
- Reduces “real” prototypes
- Iterative “what-if” analysis

**Design with Simulation**

**Executable Specifications**

**Continuous Test and Verification**
- Detects errors earlier

**Automatic Code Generation**
- Minimizes coding errors

**Automatic code generation**
Steps in MBD approach

1. Plant modeling
   - Motor, load, power electronics etc.

2. Interface modeling
   - Sensors, device drivers

3. Controller modeling
   - Field oriented control of 3 phase PM motor

4. Analysis and synthesis
   - The models created in step 1-3 are used to identify dynamic characteristics of the plant model.
   - Tuning and configuration of system

5. Validation and test
   - Offline simulation and/or real-time simulation
   - Time response of the dynamic system is investigated

6. Deployment to embedded target
   - Automatic code generation
   - Test and verification
Steps in MBD approach

- Simulation of Controller and Plant model
- Off-line development of algorithms without access to HW
- Code generation and deployment to embedded controller
- Comparison between simulation and actual implementation
Model complexity

What is a good model?
- Represent the states of interest
- Supports automatic code generation
- Execute fast
- Use existing (trusted) libraries
- Reusable across platforms

Too little  Right balance  Too much
Model complexity

- Understanding strengths and weaknesses of tools is critical when defining model scope

<table>
<thead>
<tr>
<th>Strength</th>
<th>- Solving differential equation</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>- Time domain modeling</td>
</tr>
<tr>
<td></td>
<td>- Visualization</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Weakness</th>
<th>- Target specific setup</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>- Managing system resources</td>
</tr>
<tr>
<td></td>
<td>- Time scheduling</td>
</tr>
</tbody>
</table>

- Register setup and control
- Managing System resources
- Time scheduling

- Real time control systems
- Debugging and test
- Visualization
Interfacing to Embedded Target

- **SW architecture**
  - Generic C-code needs to be tied to embedded platform
  - Utilizing strength of multiple environments
  - Graphical environment for application code development
  - “Classic” C-code for target specific control
Interfacing Embedded Target

- Two platforms – one implementation
  - Common control algorithm for *Simulation* and *Embedded system*
  - Platform specific device drivers or behavioral models
    - Debug algorithm using simulation and test on embedded platform
Drive system feedback and control

Mechanical System

- Velocity/position Loop
- Current (Torque) Loop
- Field Alignment

Electrical Circuit

- PWM
- Power Inverter

Electromechanical System

- Power system: $v_U, v_V, v_W$
- $V_m$
- $V_{DC}$

Motor

- $i_m$
- Feedback

Shaft Sensors

- $\theta$
- $\omega$

- Mechanical performance and motor protection
- Motor Efficiency and torque production
- Inverter Efficiency and protection
Motor Control Application Program

Power Inverter and Motor

Device Driver

MC Algorithm

MC 'C' code

Application code

MC Application Firmware

Device drivers

Application code

Sensors, interfaces and Device Drivers

MC Control Application Program
Shaft position sensor measures rotor magnet (flux) position – velocity is calculated.

Two/three motor currents measured (two is enough).

Clarke-Park transfer calculate Torque (Iq) and Flux (Id) components of current.

Speed loop generates \( I_q \) reference; \( I_d \) reference set to zero for PMSM.

Current loops and Field alignment generates \( a, b, c \) voltage commands.
Field oriented control
Simulation

Simulink scopes

Code debugging

```matlab
124 if theta > AbsAngle
    AbsAngle = 2^16-theta+AbsAngle;
else
    %theta < AbsAngle
    AbsAngle = AbsAngle-theta;
```
Generating C-code
Building and Linking – IAR EWB
Building and Linking

MathWorks

ARM Library
- M3/M4 support

Legacy code
- Existing code

Motor Control
- FOC functions

Device Drivers
- Functional model

Device drivers
- SW enablement package

Application code
- State machines

System resources
- Allocation & setup

Scheduling
- IRQs/RTOS

C-code

Executable

CM40x
Running target

- MATLAB GUI + embedded engine
- Streaming data back to MATLAB
Sampling domains

10 kHz

1 kHz
Summary

- Model-Based Design
  - System modeling and simulation
  - Automatic code generation
- Modeling of motor control systems
  - Electronics/mechanical
  - Interfaces
  - Control algorithm
- Interfacing generic C-code to embedded target
  - Use of different environments
  - Device drivers
- Debugging and test
  - Off-line
  - In real-time
Thank You For Watching

View Additional Webcasts at

www.analog.com/webcasts