Model-Based Design of Advanced Motor Control Systems

by Anders Frederiksen, Analog Devices, Inc.

IDEA IN BRIEF

Leveraging advanced processor functionality to facilitate ease of design has been discussed throughout recent decades. Nowadays even greater design flexibility allows engineers to use standard model-based design with MATLAB® and Simulink® to optimize motor control systems functionality and to minimize overall design time. It also enables design engineers to reuse simulation models to ensure the correct functionality and desired performance of a system in its end market application.

History	Simulation	Implementation
197x	Analog computing	Transistor amplifier
198x	Custom simulation program - Fortran	Mixed signal components
199x	Standard simulation platform	Microcontrollers – Assembly code
200x	Matlab – Simulink control schematic	Fast DSP & RISC – C code
201x	Simulink Model → Embedded C code for embedded target	

Figure 1. History and Capability

Model-based design (MBD) has been a discussion topic for decades but has only in recent years evolved into a complete design flow—from model creation to complete implementation. In the 1970s, analog computing platforms were available for simulation but control hardware implementation was done at the transistor level. Simulation tools advances through to the 2000s saw the introduction of graphical control schematic entry tools and control design tools that vastly simplified the task of complex control design and evaluation. However, the control system designer still developed the hardware control algorithm by writing C code to mirror the simulated design. Now, at the start of this decade, complete MBD allows a common control design for both simulation and hardware implementation platforms enabling complex control algorithms to be rapidly deployed on hardware platforms.

MBD is a process that uses a system model as an executable specification throughout development. This simulation-based approach offers a better understanding of design alternatives and trade-offs than traditional hardware prototype-based design methodologies, enabling you to optimize your design to meet predefined performance criteria. Rather than using complex structures and extensive software code, designers can define models with advanced functional characteristics using continuous time and discrete time building blocks. Existing C code can be integrated with standard control library blocks to maximize design efficiency. These models, used with simulation tools, can lead to rapid prototyping, software testing, and hardware-in-the-loop (HIL) simulation. Simulation enables specification discrepancies and modeling errors to be found immediately, rather than later in the design cycle. Automatic code generation eliminates the manual steps in implementing the same algorithm to run on the hardware platform. This simplifies the design process, minimizes errors in hardware design implementation, and reduces the overall time to market.

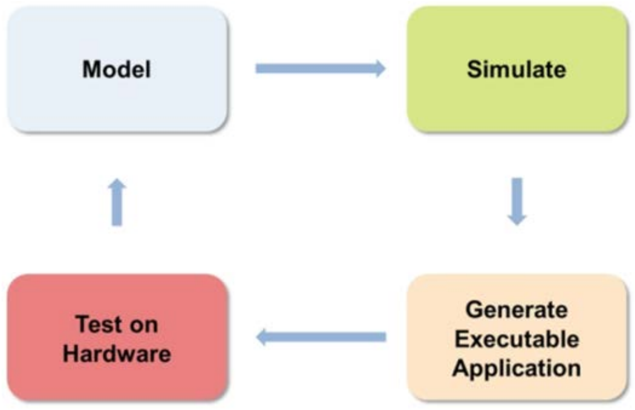


Figure 2. Design Flow of Model-Based Design (MBD)

There are multiple steps within MBD that allow optimization of individual tasks in the overall design. These tasks can be completed by different design engineers or design teams, and then combined to form the overall design and complete system. With this approach, a higher level of abstraction of the individual tasks can be applied, resulting in an overall design flow optimized for the given end application. All-in-all, MBD allows a designer to expand from more classical design schemes and move directly from model creation to simulation, code generation, and HIL test in a controlled fashion that allows incremental changes in system behavior without a complete redesign of the system.

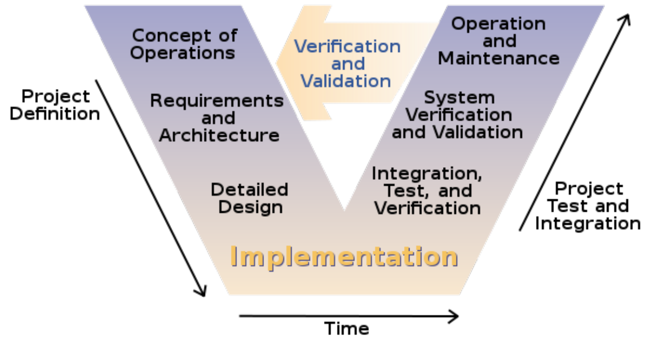

Concept of Operations
Verification and Validation
Operation and Maintenance
Project Definition
Requirements and Architecture
System Verification and Validation
Detailed Design
Integration, Test, and Verification
Project Test and Integration
Implementation
Time

Figure 3. Concept of MBD Implementation

In Figure 3, the different design phases and the scale of individual steps in the flow are visualized. These steps together describe the "standard" flow of MBD. From a motor control design perspective they are:

- Concept of operations
  - Overall functionality of the motor system
- Plant modeling/Architecture
  - Development of models of motor, load, power electronics, signal conditioning, etc.
- Controller modeling and requirements
  - Encoder-based field oriented control of three phase PM motor
- Analysis and synthesis—detailed design
  - Models created above are used to identify dynamic characteristics of the plant model
  - Tuning and configuration of the system
- Validation and test
  - Off-line simulation and/or real-time simulation
  - Investigation of time response of the dynamic system
- Deployment to embedded target—full operation
  - Automatic code generation
  - Test and verification
  - Updating controller model

Together this forms a multistep approach for aligning the overall design and allows individual control steps to be analyzed independently. Once the hardware and software specification has been completed, the complete system architecture can be setup for deployment of both specific algorithm and functionality of the overall system (see Figure 4).

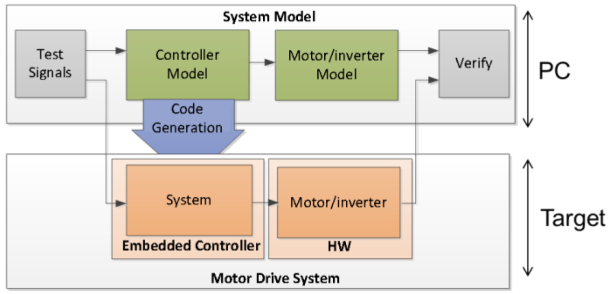
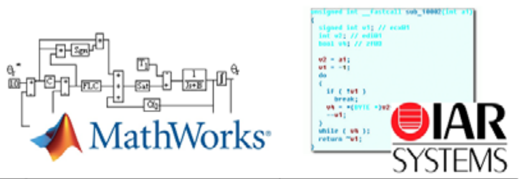


Figure 4. MBD Setup

Simulation of controller and plant models can be evaluated. Off-line development of algorithms without access to hardware can be architected and fine-tuned to meet overall system performance requirements. Initial code generation, either with existing code "reuse" or code generated by a code generation tool, can be deployed to the embedded controller to compare the system simulation on a PC with the actual implementation data on a hardware target. A designer must consider the complexity of the model when defining a correctly balanced structure for MBD. However, once a balanced concept is realized, it is possible to quickly change independent models within the design and achieve more accurate results from the entire drive system.

The experimental setup used as background for this paper is based on an ARM® Cortex™-M4 mixed-signal control processor from Analog Devices used with combined tools from IAR and MathWorks to achieve complete implementation of the MBD platform. Each of the steps discussed above has a direct link to the available tools and to the overall implementation. As seen in Figure 5, each tool chain offers a range of value. In MBD, the designer must choose how to balance the use of these tool chains with the overall value creation of the independent MBD platform.



Strength	- Solving differential equation - Time domain modeling - Visualization	- Register setup and control - Managing System resources - Time scheduling
Weakness	- Target specific setup - Managing system resources - Time scheduling	- Real time control systems - Debugging and test - Visualization

Figure 5. MathWorks and IAR System Strength

For the target platform, a real-time development environment is now in place to model, simulate, evaluate, deploy, and optimize overall system performance and capability. This is all based on MBD and the balanced selection of system parameters, resulting in best-in-class flexibility where specific optimization is needed. In this way, a scalable model of a system has been realized that facilitates use and reuse of code, either based on existing legacy code or functions, or complete new building blocks based on standard C or graphical functionality (Simulink/MATLAB models aligned with the complete simulation and deployment phase). Not only from the software perspective is it possible to change the overall setup, but also once the right device drivers for a system are developed, a designer can change system resources, hardware elements, and overall applications software for the end application or system. In addition, given the ability to control the real-time aspects of the overall system timing, optimization of system scheduling is possible directly through this environment.

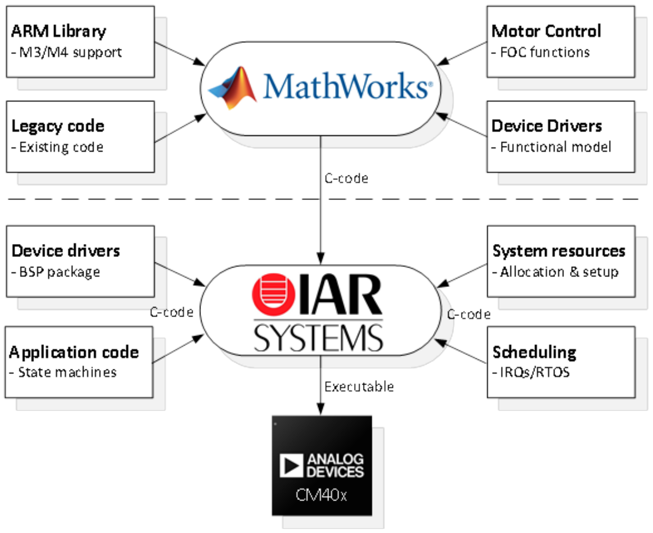


Figure 6. Deployment Environment

Taking a deeper look at the classic sketch of a drive system, one can now visualize the capability of this architecture. It is possible to optimize each of the elements in the "Drive" and focus effort on what element is most critical to the end system. This means that, for example, if protection functionality and scale is of key importance, then the focus can be placed on the mechanical system in combination with the electrical control and power system. A mix of simulated results and real-time data can be used for monitoring system behavior, and together forms a "Live" optimization. On the other hand, if noise disturbances decrease the overall efficiency level of the system, measurements of these can be used in scalable filters and observers, which minimize the hardware's noise issues to give an optimal state. Once all factors have been modeled and gathered, the final step in the deployment phase can begin and a complete implementation phase on the target system becomes a reality.

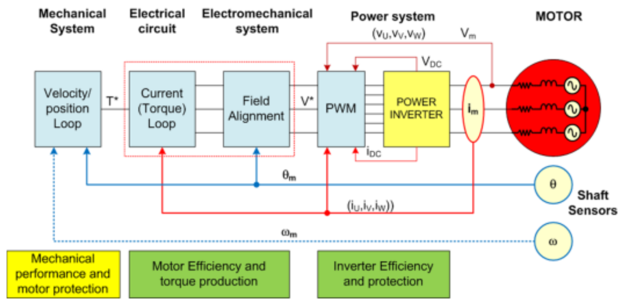


Figure 7. Sketch of a "Drive System"

Through the MBD design flow, and via MathWorks and IAR, the implementation of an overall model and the compilation of code are now possible. Each of the stages or elements in the model of the "Drive System" is now represented by a MATLAB and Simulink model that is scaled at the right level for optimal design criteria. Each of the elements in the model is based on standard toolboxes and blocksets from MathWorks and can be reused across any element in a particular design. These elements also represent the different domains of the drive system and can be fine-tuned to fully minimize the model vs. deployment error. Through real-time deployment methods and compilation in this mixed environment, it is also possible to combine existing C code written by hand with ARM Cortex-M4 optimized C code generated by Embedded Coder®, the production code generation tool for MATLAB and Simulink. This entire process allows users to reuse existing knowledge of motor control design at the right level. At this point, the IAR Embedded Work Bench can take the generated code and compile the complete project for ARM Cortex-M4, which closes the MBD implementation phase for this system.

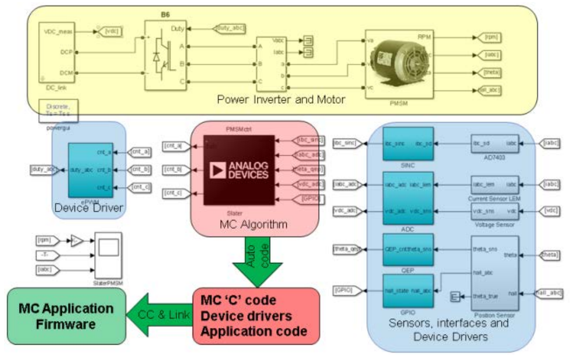


Figure 8. Implementation and Compilation

Since its inception, MBD has been questioned on its capability and functionality relative to traditional system development and on its efficient use of overall system resources. This is where strong efforts from component suppliers, simulation and deployment vendors, and tools complier providers have "merged," and results today are comparable to traditional deployment methods. Of course, any code development written for a real-time system can be created in an inefficient way, depending on the implementation method used. With MBD, profiling, crossoptimization options and strong advantages in safety critical system development can be combined, so that code development overheads are minimized and highest performance results are achieved. MathWorks offers tool qualification for use of Embedded Coder with IEC 61508, ISO 26262, and related functional safety standards.

This mix of capability is much more difficult to achieve in a standard design flow. In Figure 9, a standard FOC model is implemented on Analog Devices' ADSP-CM40x series. In this model, position and current loop feedback are executing in 15 μs, supporting real-time profiling of both the current scheme and debug facility. It also allows tracking functionality of the overall FOC scheme. Both MBD simulation results and real-time data can now be evaluated and compared with ideal system functionality in relation to the target specification. This ultimately enables a designer to constantly improve system efficiency, functionality, and performance, and to evaluate how a given element or component in the signal chain is performing against specification.

This paper has described a "new" way of architecting a motor control system with the use of MBD. Today's embedded processors must have the balance of performance, cost, and size to allow development and higher level abstractions of graphical tools, so forming the basis for highly optimized systems embracing time to market, safety, performance, and scalability. For more information on these topics and Analog Devices' capability, please go to motorcontrol.analog.com.

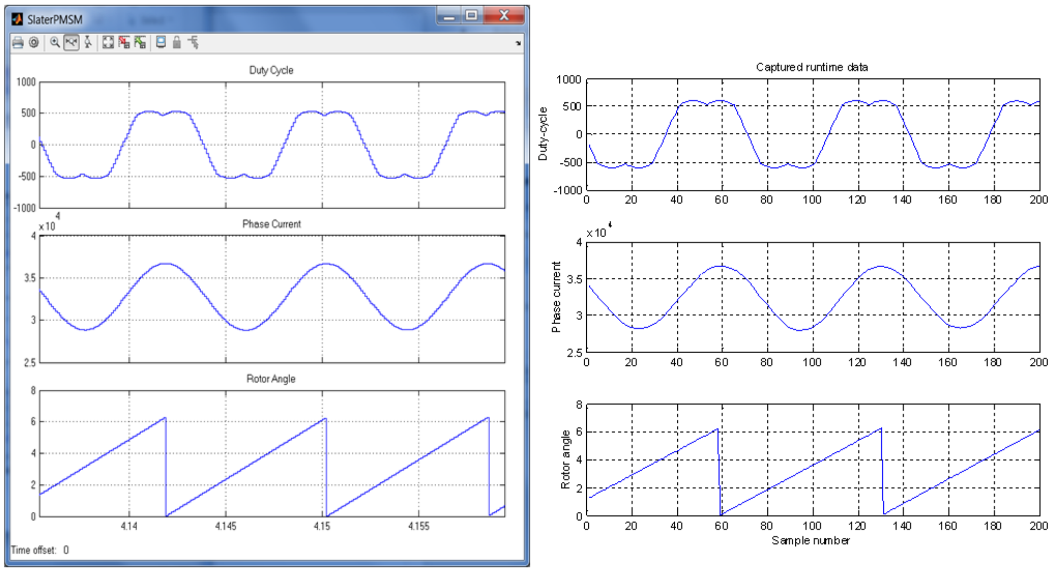


Figure 9. Simulation and Run-Time Data from Model-Based Design Build System (Processor Data vs. Simulation)

RESOURCES

Share this article on