FPGA-Based Systems Increase Motor-Control Performance

By Andrei Cozma and Eric Cigan

Introduction

Used in a wide range of industrial, automotive, and commercial applications, electric motors are controlled by drives that vary the electrical input power to control the torque, speed, and position. High-performance motor drives can increase efficiency and deliver faster, more accurate control. Advanced motor-control systems combine control algorithms, industrial networks, and user interfaces, so they require additional processing power to execute all tasks in real time. Multichip architectures are typically used to implement modern motor-control systems: a digital signal processor (DSP) executes motor-control algorithms, an FPGA implements high-speed I/O and networking protocols, and a microprocessor handles executive control.[1]

With the advent of system-on-chip (SoC) devices such as the Xilinx® Zynq All Programmable SoC, which combines the versatility of a CPU and the processing power of an FPGA, designers have the means to consolidate motor-control functions and additional processing tasks into a single device. Control algorithms, networking, and other processing intensive tasks are offloaded to the programmable logic, while supervisory control, system monitoring and diagnosis, user interface, and commissioning are handled by the processing unit. The programmable logic can include multiple control cores that run in parallel to implement multiaxis machines or multiple control systems. Having the complete controller on a single chip allows the hardware design to be simpler, more reliable, and less expensive.

In recent years, software modeling and simulation tools, such as Simulink® from MathWorks®, have allowed model-based design to evolve into a complete design flow—from model creation to implementation.[2] Transforming the way engineers and scientists work, model-based design is moving design tasks from the lab and field to the desktop. Now the entire system—including the plant and the controller—can be modeled, allowing the engineer to tune the controller's behavior before deploying it in the field. This reduces the risk of damage, accelerates system integration, and reduces dependency on equipment availability. Once the control model is complete, it can be automatically translated by the Simulink environment into C and HDL code that will be run by the control system, saving time and avoiding manual coding errors. The risk is further reduced by linking the system model to a rapid prototyping environment that allows observation of how the controller will operate in real-life conditions.

A complete development environment for achieving increased motor-control performance uses the Zynq SoC from Xilinx for implementing the controller, Simulink from MathWorks for model-based design and automatic code generation, and the Intelligent Drives Kit from Analog Devices for rapid prototyping of the drive system.

Xilinx FPGA and SoC Motor-Control Solutions

Advanced motor-control systems must execute a combination of control, communication, and user interface tasks, each of which has different processing bandwidth requirements and real-time constraints. The hardware platform chosen to implement such a control system must be robust and scalable while allowing for future system improvements and expansion. The Zynq All Programmable SoC fulfills these requirements by combining a high-performance processing system with programmable logic, as shown in Figure 1. This combination delivers superior parallel-processing power, real-time performance, fast computation, and versatile connectivity. The SoC integrates two Xilinx analog-to-digital converters (XADC) for monitoring the system or external analog sensors.

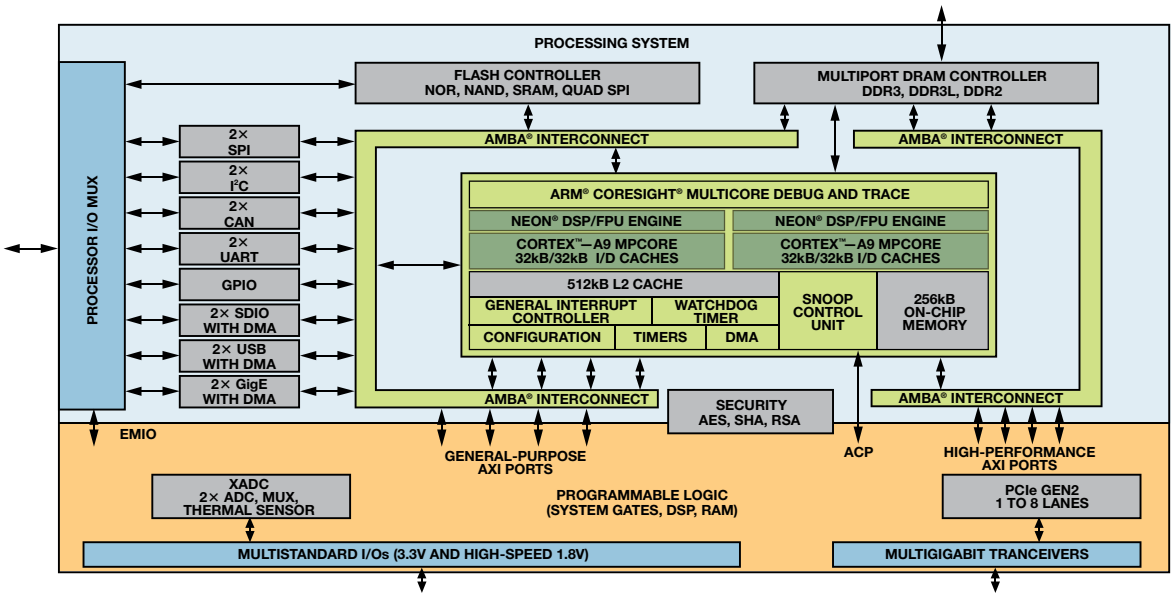

Figure 1. Xilinx Zynq SoC block diagram.

The processing side of the Zynq consists of a dual-core ARM Cortex-A9 processor, a NEON coprocessor, and floating-point extensions that accelerate software execution. The processing system addresses tasks such as supervisory control, motion control, system management, user interface, and remote maintenance functions that are well-suited to software implementation. Embedded Linux or real-time operating systems can be deployed to take advantage of the system's capabilities. The self-contained processor can be used without the need to configure the programmable logic. This allows software developers to write code in parallel with hardware engineers who design the FPGA fabric.

On the programmable logic side, the device has up to 444,000 logic cells and 2200 DSP slices that supply massive processing bandwidth. The FPGA fabric is scalable, so a user can choose from a small device with 28,000 logic cells all the way up to a high-end device, which can tackle the most challenging signal-processing applications. Five AMBA-4 AXI high-speed interconnects tightly couple the programmable logic to the processing system, giving the equivalent of more than 3000 pins of effective bandwidth. The programmable logic is suitable for implementing time critical, processing intensive tasks such as real-time industrial Ethernet protocols, and it can accommodate multiple control cores that run in parallel for multiaxis machines or multiple control systems.

Xilinx All Programmable SoC-based solutions and platforms meet the critical timing and performance requirements posed by today's complex control algorithms such as field-oriented control (FOC) and complex modulation schemes such as the regenerative pulse frequency modulator[3] designed by Xilinx and Qdesys.

Model-Based Design Using Simulink from MathWorks

Simulink is a block diagram environment for multidomain simulation and model-based design that is well-suited to simulating systems that include control algorithms and plant models. Motor-control algorithms regulate speed, torque, and other parameters, often for precision positioning. Evaluating control algorithms using simulation is an effective way to determine the suitability of motor-control designs and reduce the time and cost of algorithm development before committing to expensive hardware testing. Figure 2 depicts an efficient workflow for designing a motor-control algorithm:

- Build accurate controller and plant models, often from libraries of motors, drive electronics, sensors, and loads
- Simulate system behavior to verify that the controller is performing as expected
- Generate C code and HDL for real-time testing and implementation
- Test control algorithms using prototyping hardware
- Deploy the controller onto the final production system once the control system has proven to be satisfactory through simulation and testing on prototyping hardware

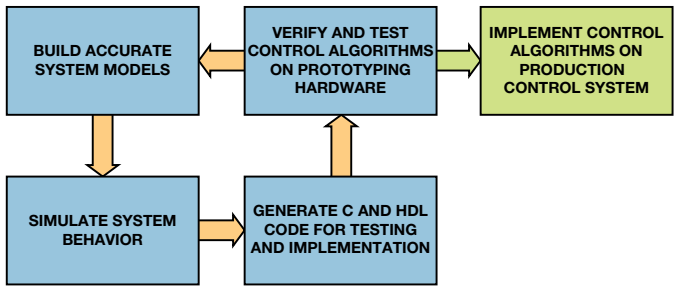

Figure 2. Workflow for motor-control algorithm design.

MathWorks products including the Control System Toolbox,™ SimPowerSystems,™ and Simscape™ provide industry-standard algorithms and applications for systematically analyzing, designing, and tuning linear control systems, as well as component libraries and analysis tools for modeling and simulating systems spanning mechanical, electrical, hydraulic, and other physical domains. These tools provide the means to create high-fidelity plant and controller models that can verify the behavior and performance of the control system before moving to the physical implementation. The simulation environment is the perfect place to verify functionality corner cases and extreme operating conditions to ensure that the controller is prepared for such situations, and its real-life operation will be safe for both equipment and operating personnel.

Once the control system is fully verified in the simulation environment using the embedded coder and the HDL coder tools, it can be translated into C code and HDL and deployed on prototyping hardware for testing and to the final production system afterwards. At this point software and hardware implementation such as fixed-point and timing behavior requirements are specified. Automatic code generation helps to reduce the time needed to move from concept to actual system implementation, eliminates coding errors, and ensures that the actual implementation matches the model. Figure 3 depicts the real-life steps needed to model a motor controller in Simulink and transfer it to the final production system.

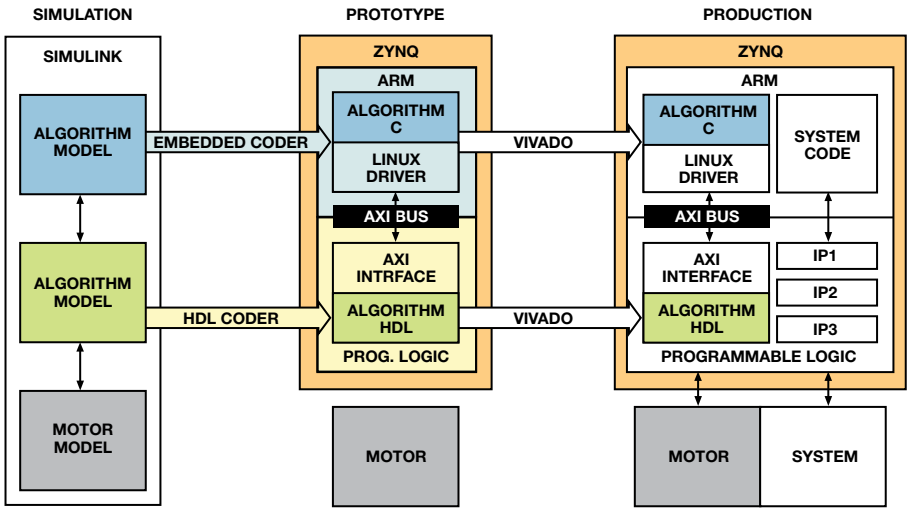

Figure 3. Path from simulation to production.

The first step is to model and simulate the controller and the plant in Simulink. At this stage, the controller algorithm is partitioned into blocks that will be implemented in software, and blocks that will be implemented in programmable logic. Once the partitioning and the simulation are complete, the controller model is converted into C code and HDL using the embedded coder and HDL coder. A Zynq-based prototyping system verifies the performance of the control algorithm and helps further tune the controller model before moving to the production stage. In the production stage, the automatically generated C code and HDL are integrated into the complex production system framework. This workflow ensures that once the control algorithm reaches the production stage it is fully verified and tested, thus providing high confidence in the system robustness.

Rapid Prototyping with the Analog Devices Intelligent Drives Kit

Choosing the right prototyping hardware is a major step in the design process. The Analog Devices Intelligent Drives Kit enables rapid, efficient prototyping. Combining the Zynq-7000 All Programmable SoC ARM dual-core Cortex-A9 + 28 nm programmable logic with the latest generation Analog Devices high-precision data converters and digital isolation, the Avnet Zynq-7000 All Programmable SoC/Analog Devices Intelligent Drives Kit enables high-performance motor control and dual gigabit Ethernet industrial networking connectivity. The kit comes with an Avnet ZedBoard 7020 baseboard and ADI's AD-FMCMOTCON1-EBZ module, which is a complete drive system providing efficient control for multiple motor types. In addition, the kit can be extended with ADI's AD-DYNO1-EBZ dynamometer drive system, a dynamically adjustable load that can be used to test real-time motor-control performance. The AD-FMCMOTCON1-EBZ module consists of a controller and a drive board, as shown in Figure 4.

The controller board is a mixed-signal FPGA mezzanine card (FMC), designed to connect to any Xilinx FPGA or SoC platforms with low pin count (LPC) or high pin count (HPC) FMC connectors. It features:

- Current and voltage measurement using isolated ADCs
- An isolated Xilinx XADC interface
- Fully isolated digital control and feedback signals
- Hall, differential Hall, encoder, and resolver interfaces
- 2-Gb Ethernet PHYs to enable high-speed industrial communication protocols such as EtherCAT, ProfiNET, Ethernet/IP, or Powerlink
- FMC signals voltage adaptation interface for seamless operation on all FMC voltage levels

Isolation, a critical aspect of any motor-control system, is required to protect the controller as well as the user. Full isolation of the analog and digital signals on the controller board ensures that the FPGA platform is always protected from dangerous voltages that can arise on the motor drive side.

The drive board contains all the power electronics needed to drive the motors as well as current and voltage sensing and protection circuits. The board features:

- Drives BLDC (brushless dc)/PMSM (permanent-magnet synchronous motor)/brushed dc/stepper motors in the 12-V to 48-V range with 18-A maximum current
- Dynamic braking capability and integrated overcurrent and reverse voltage protection
- Phase-current measurement using isolated ADCs; programmable-gain amplifiers maximize the current measurement input range
- Provides dc bus voltage, phase current, and total current feedback signals to the controller board
- Integrated BEMF zero-crossing detection for sensorless control of PMSM or BLDC motors

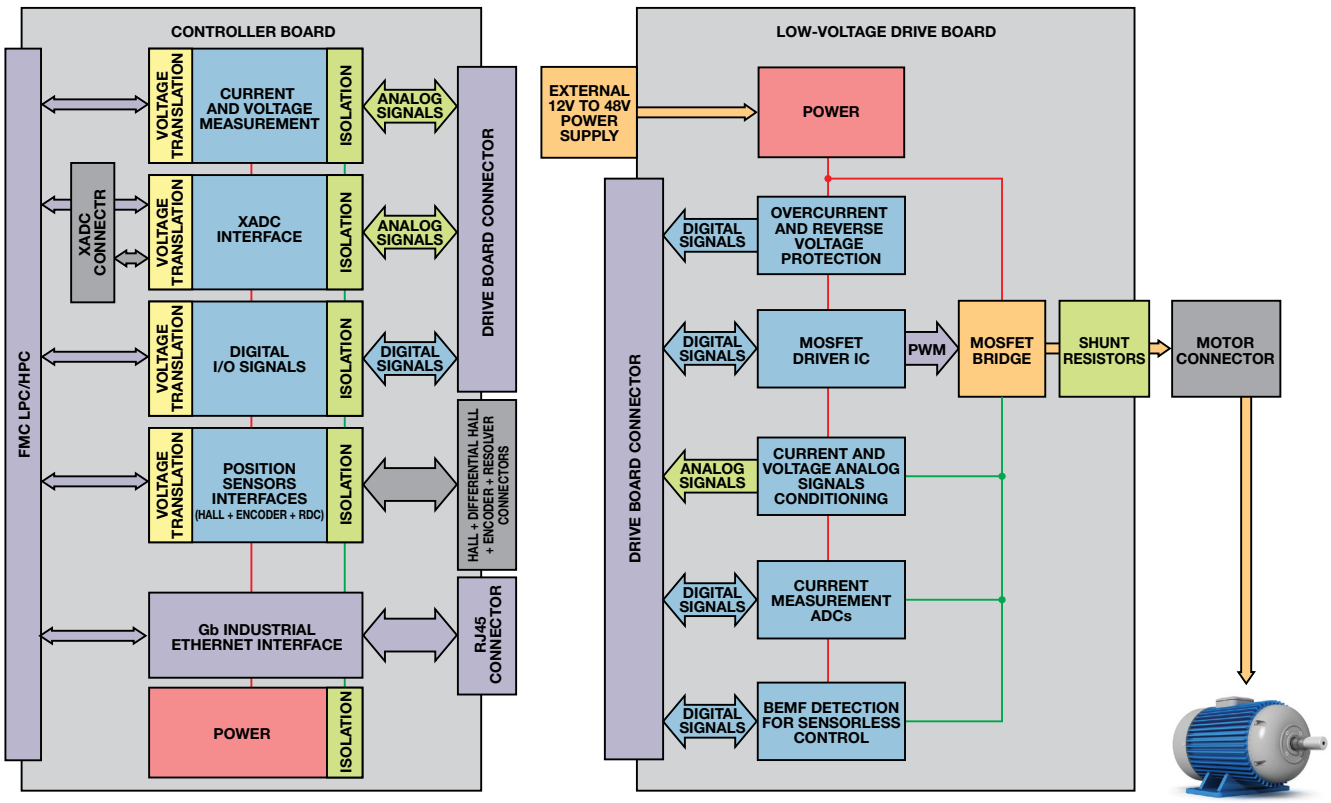

Figure 4. AD-FMCMOTCON1-EBZ block diagram.

The dynamometer, which is a dynamically adjustable load that can be used to test real-time motor-control performance, consists of two BLDC motors directly coupled through a rigid connection. One of the BLDC motors acts as a load and is controlled by the dynometer's embedded control system; the second motor is driven by the ADI Intelligent Drives Kit, as shown in Figure 5. The system, equipped with a user interface that displays information about the load current and speed, allows different load profiles to be set. External control can be achieved by using the Analog Discovery USB oscilloscope for load signal capture and control directly from MATLAB® using the MathWorks Instrument Control Toolbox.™

The performance of any motor-control system is greatly influenced by the quality of motor current and voltage measurements. By using high-performance analog signal conditioning components and ADCs, the ADI Intelligent Drives Kit provides precise current and voltage measurements. The measurement paths are divided between the controller and the drive board as shown in Figure 6.

The phase currents are sensed by measuring the voltage across shunt resistors. Two possible measurement paths aim to get the best measurement accuracy depending on whether the ADC is close to the shunt resistor or not. When the ADC is close to the shunt resistor, the signal path is very short and less prone to noise coupling. The small differential voltage on the shunt resistor is measured directly with the AD7401 isolated Σ-Δ modulator without the need for extra interfacing and signal conditioning circuitry. When the ADC is far from the shunt resistor, the signal path is long and prone to noise coupling, especially from power supply switching noise and the motor. Special care must be taken to ensure that the PC board traces and signal conditioning circuitry between the ADC and the shunt resistor are properly shielded. The small differential voltage on the shunt resistor is amplified on the drive board with the AD8207 difference amplifier, which is placed close to the shunt resistor to avoid noise coupling. The signal is amplified from a ±125-mV full-scale input range to a ±2.5-V range to minimize the effect of the coupled noise. The amplified signal goes through another amplification stage using the AD8251 programmable-gain instrumentation amplifier (PGIA), ensuring that the ADC always receives input signals that are properly scaled to fit the input range. The amplified analog signals go through the connector to the controller board. The connector includes shielding for each analog signal to mitigate noise coupling. The analog signals coming from the drive board are shifted back to the AD7401 input range using the ADA4084-2 operational amplifier.

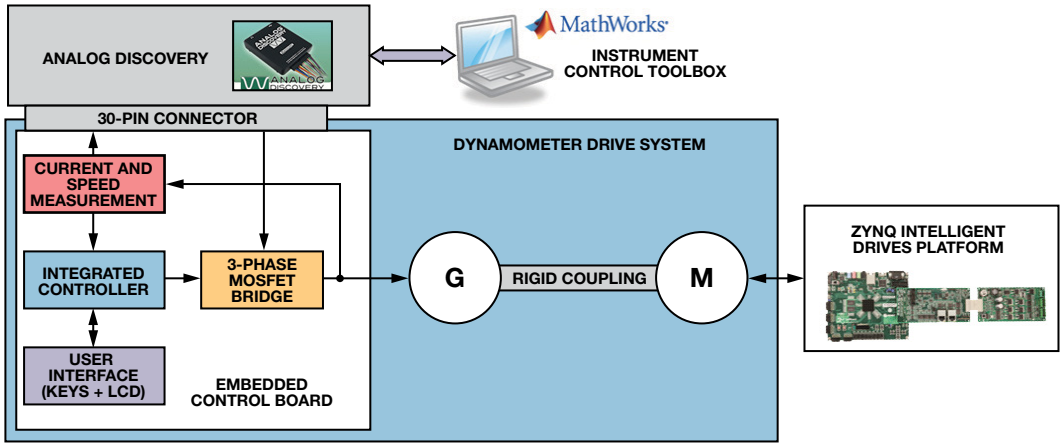

Figure 5. Dynamometer drive system.

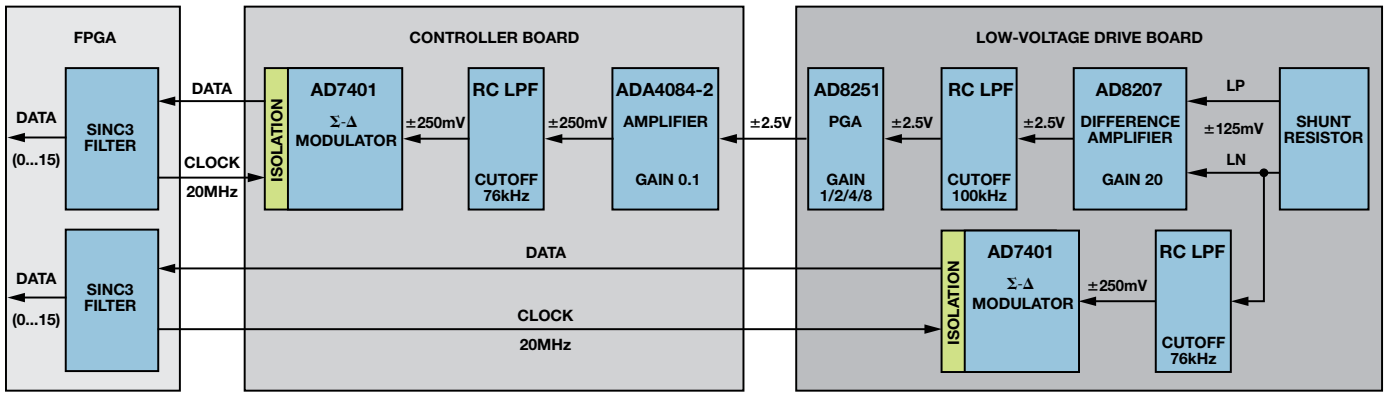

Figure 6. Phase-current signal chain.

The most important part in the current and voltage feedback signal chain is the AD7401A second-order isolated Σ-Δ modulator. This high-performance ADC features 16-bit resolution with no missing codes, 13.3 effective number of bits (ENOB), and 83-dB SNR. The 2-wire digital interface includes a 20-MHz clock input and a 1-bit digital bitstream output. The ADC output is reconstructed using a sinc^3 digital filter. A filter model and HDL implementation are provided in the data sheet for a 16-bit output and a 78-kHz sampling rate. The output resolution and sampling rate can be controlled by changing the filter model and decimation. While the 78-kHz sampling rate might be good enough for many applications, some situations require a higher rate. In these cases, filter banks, as shown in Figure 7, can be used to increase the sampling rate of the system up to 10 MSPS of true 16-bit data. The filter bank contains n sinc^3 filters with sampling clocks that are delayed by multiples of T, which is the sinc^3 filter propagation time divided by n. The data selector outputs the ADC code with a periodicity equal to T.

Phase-current measurements can be also performed by the Zynq XADC. The XADC signal measurement chain uses the entire path of the regular measurement chain and adds a Sallen-Key analog reconstruction filter after the AD7401 Σ-Δ modulator. This filter is implemented on the controller board using AD8646 operational amplifiers, as shown in Figure 8. The combination of the isolated Σ-Δ modulator and the analog reconstruction filter provides a convenient, low-cost way to achieve analog isolation of the XADC input signals without compromising the quality of the measurement.

The Analog Devices Intelligent Drives Kit comes with a set of Simulink controller models, a complete Xilinx Vivado framework, and an ADI Linux infrastructure, allowing a user to go through all the steps needed to design a motor-control system, starting with simulation, going through prototyping, and finishing with the production system implementation.

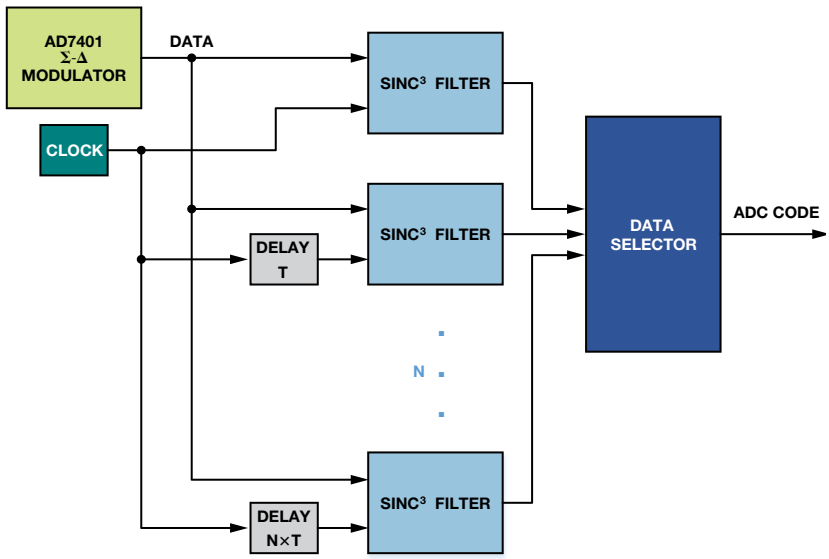

Figure 7. Filter bank.

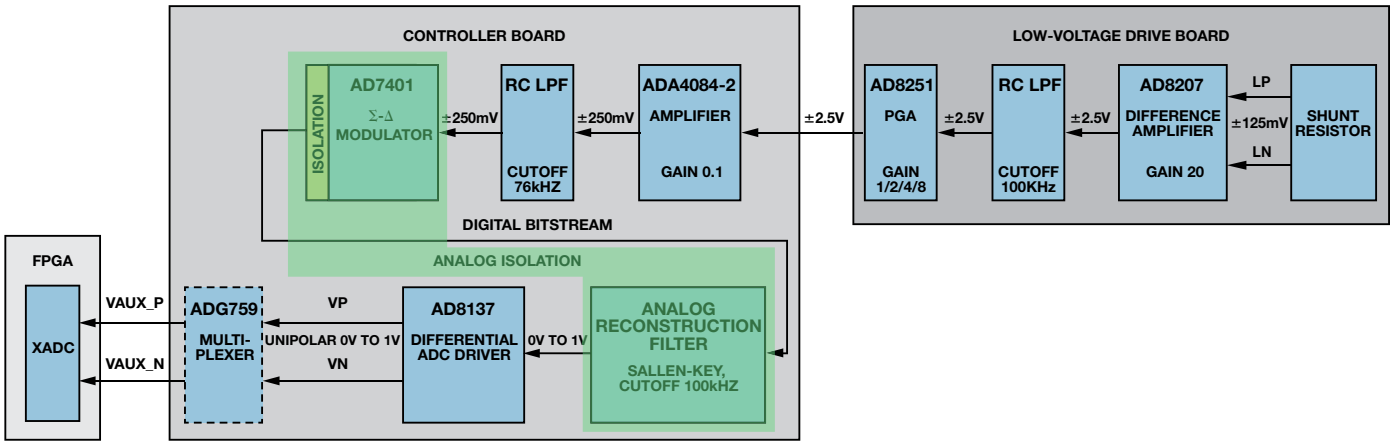

Figure 8. XADC signal measurement chain.

Two controller models—a six-step controller and a PMSM field-oriented controller—can be used to start the design process. Figure 9 shows the top-level views of these two controllers. The six-step controller implements a trapezoidal controller for BLDC motors; the FOC controller provides an FOC core for integration within the control system.

The plant and controller models are created in the simulation stage, and the behavior of the complete system is simulated to verify that the controller is performing as expected. The controller model is partitioned into components that will be implemented in C code and HDL, and constraints such as timing, fixed-point implementation, sampling rates, and loop times are specified to ensure that the controller model behaves as it will in the hardware implementation. Figure 10 shows the partitioning of the six-step controller between software and HDL.

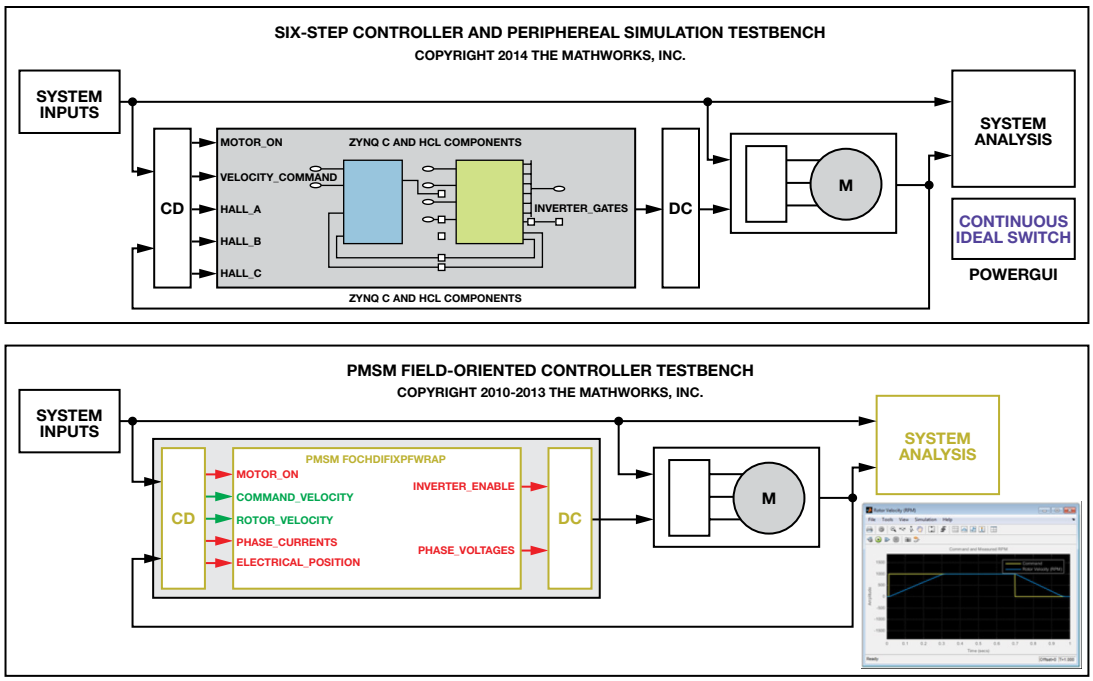

Figure 9. Simulink controller models.

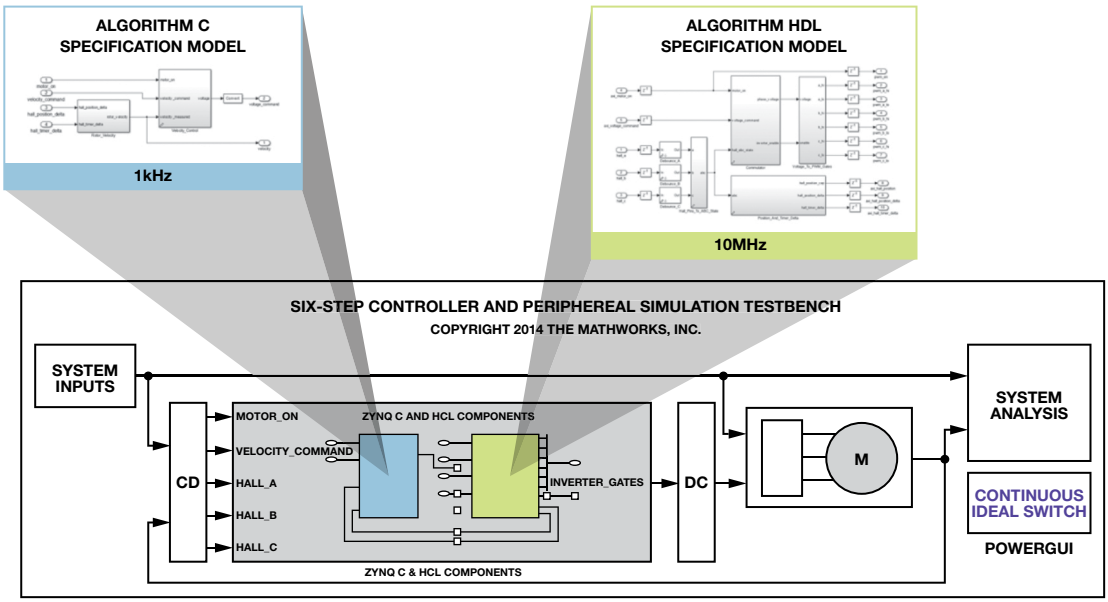

Figure 10. Controller partitioning in C code and HDL.

Once the controller is fully verified in simulation, the next step is to prototype it on the hardware platform. The Zynq SoC guided workflow generates the C code and HDL from the Simulink model partitioned into subsystems targeting the ARM core and the programmable logic. With this workflow, the HDL coder generates the HDL that targets the programmable logic, while the embedded coder generates the C code targeting the ARM. The MathWorks Zynq support package enables the generation of the ARM executable consisting of algorithmic C code from models, which interfaces to the AXI bus, as well as bitstream generation consisting of HDL code from models, which interfaces to programmable logic pins and AXI bus. Figure 11 shows the controller implementation and the relationship with the ADI Intelligent Drive hardware.

Once the bitstream and executable are loaded into the hardware, operational testing of the controller can begin. Hardware-in-the-loop (HIL) testing is performed using an Ethernet link between Simulink and the embedded system running an open-source Linux OS. Motor parameters such as shaft speed can be captured in Simulink and compared against simulation results to make sure that the physical system implementation matches the model. Once the control algorithm testing is complete, the controller can be transferred to the production system.

Together with the Intelligent Drives Kit, Analog Devices provides a complete Vivado framework and a Linux infrastructure that can be used for both prototyping and final production. Figure 12 shows the Zynq Infrastructure that supports the Intelligent Drives Kit. This high-level diagram shows how the ADI reference design is partitioned on a Xilinx Zynq SoC. The programmable logic implements the IP cores for interfacing with the ADCs, position sensors, and the motor drive stage. The HDL, generated by the HDL coder to represent the motor-control algorithm, is integrated into the Analog Devices IP. All the IPs have low-speed AXI-Lite interfaces for configuration and control, and high-speed AXI-Streaming interfaces that allow them to transfer real-time data through DMA channels to the software level. The high-speed Ethernet interfaces can be implemented using either the hard MAC peripherals of the ARM processing system or the Xilinx Ethernet IPs in the programmable logic.

The ARM Cortex A9 processing system runs Ubuntu Linux, provided by Analog Devices. This includes the Linux IIO drivers needed to interface with the Analog Devices Intelligent Drive hardware, the IIO Oscilloscope (Scope) user space application for monitoring and control, a libiio server that allows real-time data acquisition and system control over TCP, clients running on a remote computer, and optional user applications that incorporate C code generated by the embedded coder.

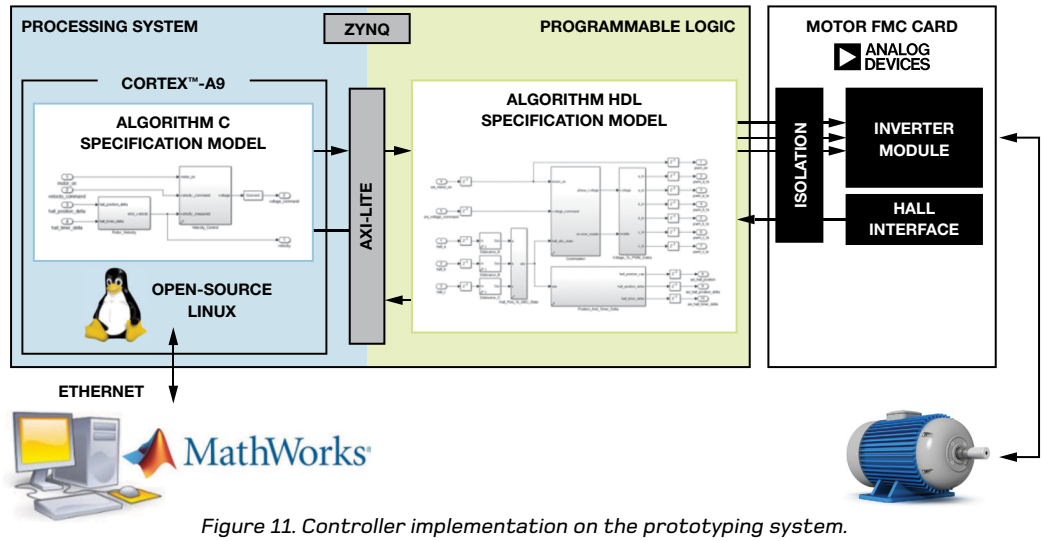

Figure 11. Controller implementation on the prototyping system.

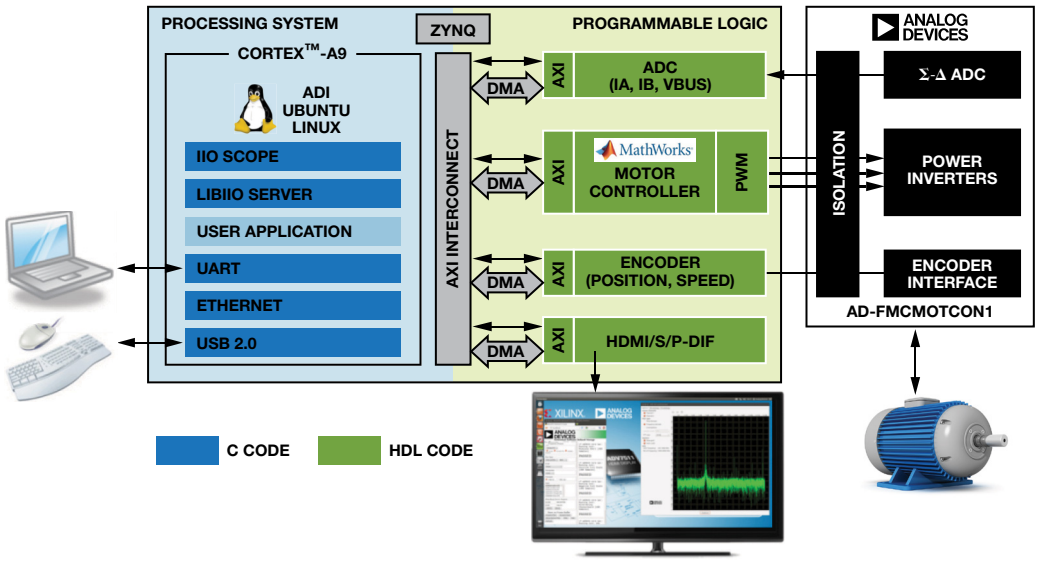

Figure 12. ADI Linux infrastructure.

All the ADI Linux drivers are based on the Linux Industrial I/O (IIO) subsystem, which is now included in all mainline Linux kernels. The IIO Scope, an open-source Linux application developed by Analog Devices that runs on the dual ARM Cortex A9s inside the Xilinx Zynq, has the ability to display real-time data acquired from any Analog Devices FMC card connected to the Xilinx Zynq platform. The data can be displayed either in the time domain, frequency domain, or as a constellation plot. Different popular file formats, such as comma separated values or .mat Matlab files, are supported to save the captured data for further analysis. The IIO Scope provides a graphical user interface for changing or reading back the configuration of the Analog Devices FMC cards.

The libiio server allows real-time data acquisition and system control over TCP together with clients running on a remote computer. The server runs on an embedded target under Linux and manages real-time data exchange over TCP between the target and a remote client. An IIO client is available as a system object to be integrated in native MATLAB and Simulink applications. An HDMI output is used to display the Linux interface on a monitor while a keyboard and mouse can be connected to the system on a USB 2.0 port.

The Linux software and HDL infrastructure provided by ADI for the Intelligent Drives Kit, together with the tools provided by MathWorks and Xilinx, are ideal for prototyping motor-control applications. They also contain production-ready components that can be integrated into the final control system, thus helping to reduce the time and cost needed to move from concept to production.

Conclusion

This article illustrates the requirements and trends of the modern FPGA-enabled motor-control systems and the tools and systems that MathWorks, Xilinx, and Analog Devices bring to the market in order to meet these constraints and help drive towards more efficient and accurate motor-control solutions. By combining the model-based design and automatic code generation tools from MathWorks with the powerful Xilinx Zynq SoCs and the Analog Devices isolation, power, signal conditioning, and measurement solutions, the design, verification, testing, and implementation of motor drive systems can be more effective than ever, leading to improved motor-control performance and reduced time to market. The Analog Devices Intelligent Drives Kit paired with the Avnet Zynq-7000 All Programmable SoC provides a great prototyping environment for the motor-control algorithms designed using Simulink from MathWorks. The Intelligent Drives Kit comes with a set of reference designs[4] intended to give a starting point for anyone who wants to evaluate the system and help kick-start any new motor control project.

References

1. Hill, Tom. "Motor Drives Migrate to Zynq SoC with Help from Matlab." *Xcell Journal,* Issue 87, Second Quarter 2014.
2. O'Sullivan, Dara, Jens Sorensen, and Anders Frederiksen. "Model Based Design Tools in Closed Loop Motor Control." *PCIM Europe,* 2014.
3. Corradi, Dr. Giulio. "Fpga High Efficiency, Low Noise Pulse Frequency Space Vector Modulation—Part I." *EDN Network,* October 04, 2012.
4. AD-FMCMOTCON1-EBZ User Guide.

Andrei Cozma

Andrei Cozma [andrei.cozma@analog.com] is an engineering manager for ADI, supporting the design and development of system level reference designs. He holds a B.S. degree in industrial automation and informatics and a Ph.D. in electronics and telecommunications. He has been involved in the design and development of projects from different industry fields such as motor control, industrial automation, software-defined radio, and telecommunications.

Eric Cigan

Eric Cigan [Eric.Cigan@mathworks.com] is in MathWorks technical marketing, supporting SoC and FPGA design workflows. Prior to joining MathWorks, he held technical marketing roles at MathStar, AccelChip, and Mentor Graphics. Eric earned a B.S. and M.S. degree in mechanical engineering from the Massachusetts Institute of Technology.