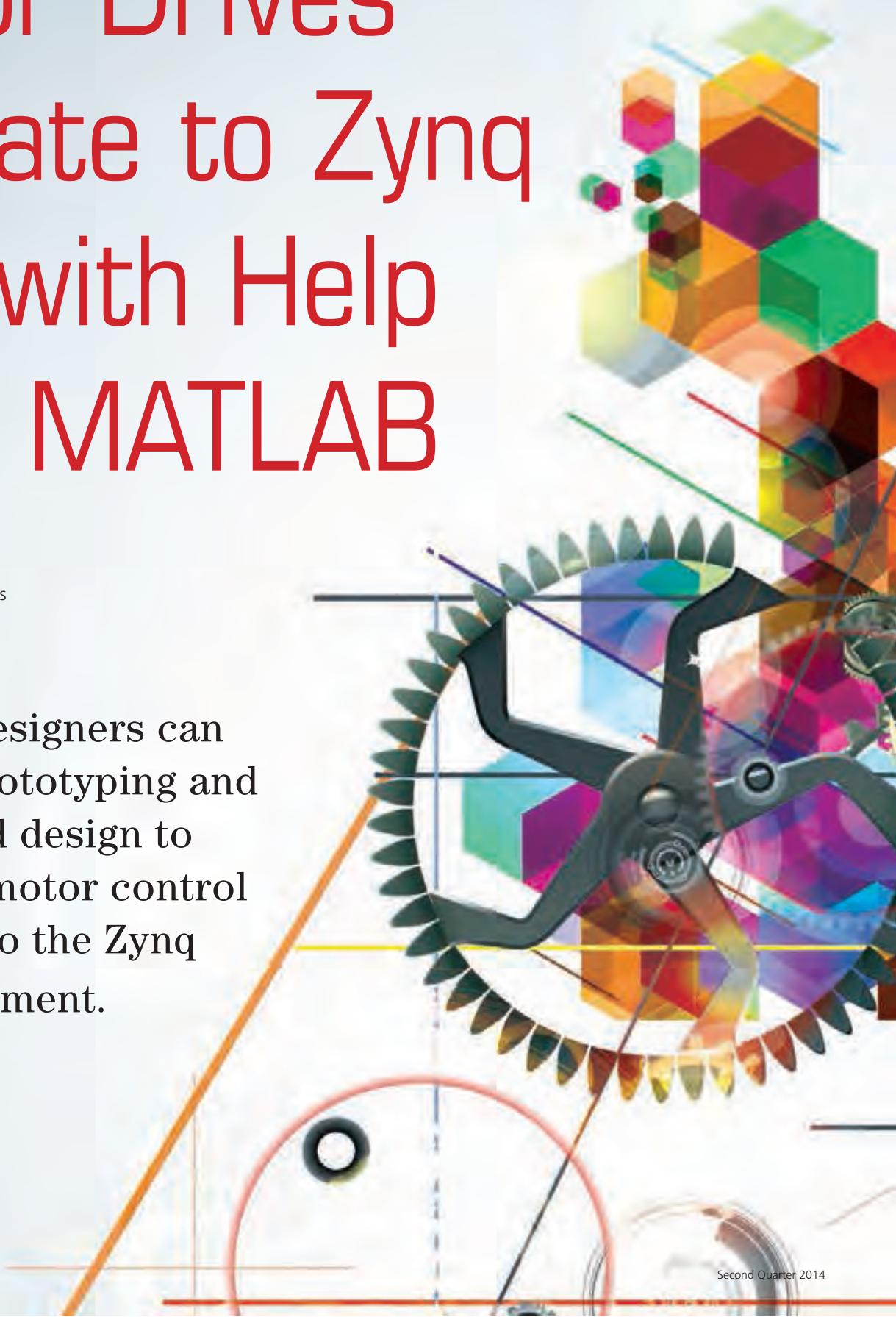
Motor Drives Migrate to Zynq SoC with Help from MATLAB

by Tom Hill
Senior Manager, DSP Solutions
Xilinx, Inc.
tom.hill@xilinx.com



Industrial designers can use rapid prototyping and model-based design to move their motor control algorithms to the Zynq SoC environment.

Since the 1990s, developers of motor drives have been using a multichip architecture to implement their motor control and processing requirements. In this architecture, a discrete digital signal-processing (DSP) chip executes motor control algorithms, an FPGA implements high-speed I/O and networking protocols, and a discrete processor handles executive control. With the advent of the Xilinx® Zynq®-7000 All Programmable SoC, however, designers have the means to consolidate these functions into a single device while integrating additional processing tasks. The reduction in parts count and complexity makes it possible to lower system cost while improving performance and reliability.

But how can drive developers evolve their established design practices to leverage the Zynq SoC?

Industrial designers have long embraced model-based design for the development of custom motor algorithms on DSP chips through the use of simulation and C-code generation. Now, a new workflow from MathWorks—developed in conjunction with Xilinx—extends model-based design to the processing system and programmable logic available with the Zynq-7000 All Programmable SoC.

ZYNQ SOCS FOR MOTOR CONTROL

Today's advanced motor control systems are a combination of control algorithms and industrial networks, including EtherCAT, Profinet, Powerlink and Sercos III, that draw processing bandwidth from the computing resources. Moreover, other requirements are converging into the control system including motion-control layers, PLC layers, diagnostic layers and user interfaces for commission and maintenance or remote monitoring. These requirements translate into logical and physical partitions with elements that fit naturally into the processing systems while other elements best fit into the hardware-assisted offloading and acceleration.

The hardware platform you select should provide a robust and scalable system. Xilinx's Zynq SoCs fulfill these requirements by supplying a high-performance processing system to address the networking, motion, soft-PLC, diagnostic and remote-maintenance functions combined with programmable logic to accelerate performance-critical functions in hardware. On the processing side, the Zynq SoC

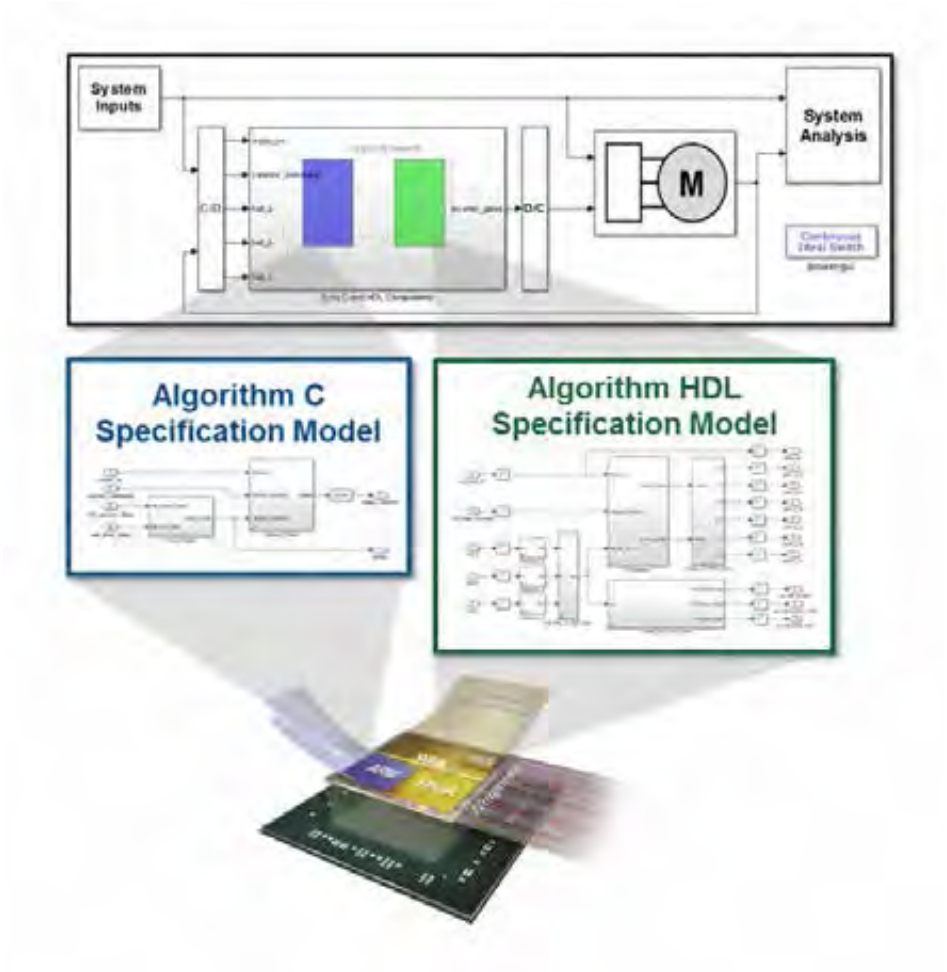


Figure 1 – MathWorks' workflow targeting the Zynq SoC, using C and HDL code generation

combines a dual-core ARM® Cortex™-A9 processing system with a NEON coprocessor and floating-point extensions to accelerate software execution. On the programmable logic side, the device has up to 444,000 logic cells and 2,200 DSP48 slices that supply massive processing bandwidth. Five high-throughput AMBA®-4 AXI high-speed interconnects tightly couple the programmable logic to the processing system with the equivalent of more than 3,000 pins of effective bandwidth.

Table 1 lists the processing performance that Zynq SoC devices can achieve.

PLANT AND MOTOR MODELING USING SIMULINK AND THE CONTROL SYSTEMS TOOLBOX

Modern control algorithms have system times and system variables that span several orders of magnitude, making hardware/software partitioning a daunting, time-consuming and iterative task. Figure 2 depicts a typical electrical drive. The power source is normally 50 to 60 Hz and is rectified to achieve a continuous voltage (DC). This DC voltage is then converted into a variable frequency that controls the power stage that feeds the motor terminals. The controller also must read the motor's basic variables including current and voltages. It likewise must read or establish the shaft position including its speed and handling commands originating from the communication network or supervising controller.

Simulink® provides a block-diagram environment for multidomain system simulation and model-based design that is well suited to simulating systems that include control algorithms and plant models. MathWorks products such as the Control Systems Toolbox provide a variety of "apps" based on widely used methods of systematically analyzing, designing and tuning control systems modeled in Simulink. Performing system modeling in Simulink can accelerate development of motor control systems while reducing risk in the following ways:

- **Reduces risk of damage** – Simulation allows thorough examination of new control system algorithms before they are tested on production hardware, where there are risks of damaging drive electronics, motors and other system components.
- **Accelerates system integration** – Support staff must integrate new control system algorithms into the

Elements	Performance (up to)
Processors (each)	1 GHz
Processors (aggregate)	5,000 DMIPs
DSP (each)	741 MHz
DSP (aggregate)	2,662 GMACs
Transceivers (each)	12.5 Gbps
Transceivers (aggregate)	200 Gbps
Software acceleration	10x

Table 1 – Processing performance of the Zynq SoC

production system, meaning that deploying new controllers can consume their limited time and can make the deployment a protracted process.

- **Reduces dependency on equipment availability** – The production environment itself may not be available, such as in cases where custom drive electronics or electric motors are under development or are not located where control system designers can access them.

Given these factors, simulation provides an excellent alternative to testing on production hardware. Simulation environments such as Simulink provide a framework for creating plant models from preexisting libraries of building blocks of electromechanical components for the evaluation of new control system architectures against plant models.

Risk to the schedule is further reduced by linking the system model to a rapid-prototyping environment as well as the final production system. The rapid-prototyping flow enables algorithm developers to prototype without having to depend on hardware designers. Instead they use a platform-specific support package in a highly automated process that deploys the hardware and software components of the system to a design template that can be compiled to a specific hardware development platform. The hardware and software design teams can reuse these same hardware and software components in the final production systems without modification to accelerate development and reduce errors.

RAPID PROTOTYPING USING THE AVNET INTELLIGENT DRIVES KIT

Designers can pair the Avnet Zynq-7000 AP SoC / Analog Devices Intelligent Drives Kit with Simulink and the Zynq SoC workflow for a complete rapid-prototyping system for motor control applications. This kit combines the Zynq SoC with the latest generation of Analog Devices' high-precision data converters and digital isolation. The kit enables high-performance motor control and dual Gigabit Ethernet industrial networking connectivity (*http://www.xilinx.com/products/boards-and-kits/1-490M1P.htm*).

It comes with an Avnet ZedBoard 7020 baseboard; Analog Devices' AD-FMCMOTCON1-EBZ module, which is capable of driving brushless DC and stepper motors with a 24-volt external power supply (included with the kit); and a 24-V BLDC motor rated for 4,000 RPM and equipped with Hall-effect sensors and a 1,250-CPR indexed encoder. Also included are a Zynq SoC reference design of field-oriented control and Analog Devices' Ubuntu Linux framework including drivers, application software and source code.

EXAMPLE: TRAPEZOIDAL MOTOR CONTROL

Let's apply this workflow to the trapezoidal motor control system in Figure 1 using simulation in Simulink to evaluate a controller with a simulated plant, then prototype the controller using the Intelligent Drives Kit. As a final step, we will validate the Simulink model using results from hardware testing.

In this example, we will use the kit to drive an inertial load in the form of an aluminum disc, with a basic trapezoidal controller. The controller's main components are as follows:

- Hall-effect sensor – detects the motor position
- Velocity estimator – computes rotor velocity based on the sensor signal
- Six-step commutator – computes the phase voltages and inverter enable signals based on rotor position and velocity
- Pulse-width modulation (PWM) – drives the controller outputs out through the drive circuitry

We start by using a behavioral, control-loop model of the system suited to control-loop analysis. First we will evaluate the model in simulation by subjecting it to a pulse test, commanding a rotational rate of 150 radians per second for 2 seconds and then returning to a stop. Through tuning of the control loop's proportional-integral (PI) controller gains, we can achieve a settling time of 1.2 seconds with negligible overshoot (control-loop simulation results appear as the purple-shaded signal in Figure 3; details on this example are available at *mathworks.com/zidk*).

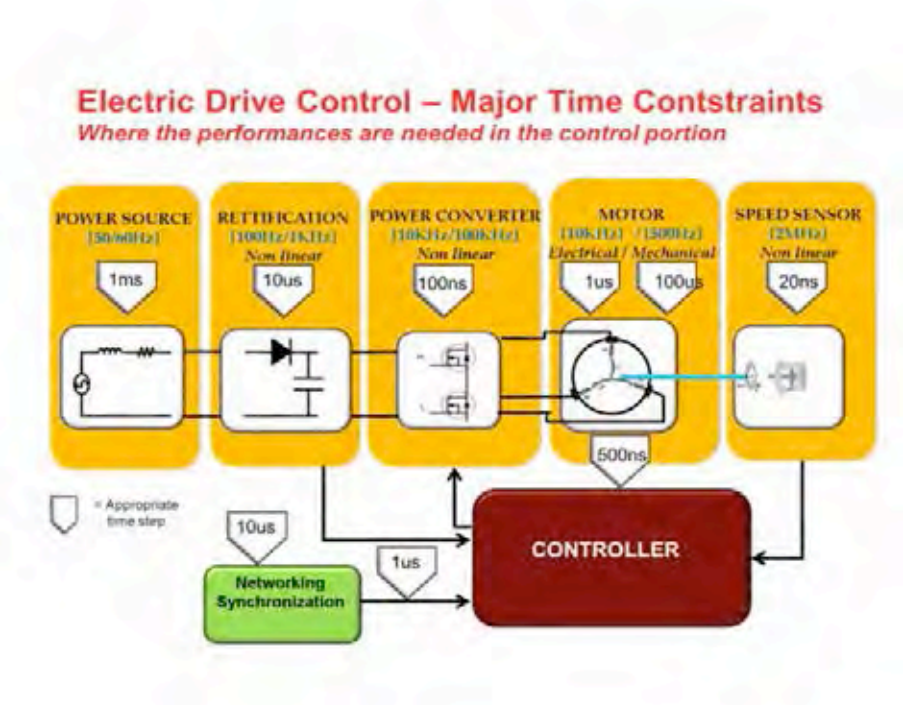


Figure 2 – Major time constraints of electrical drive controllers

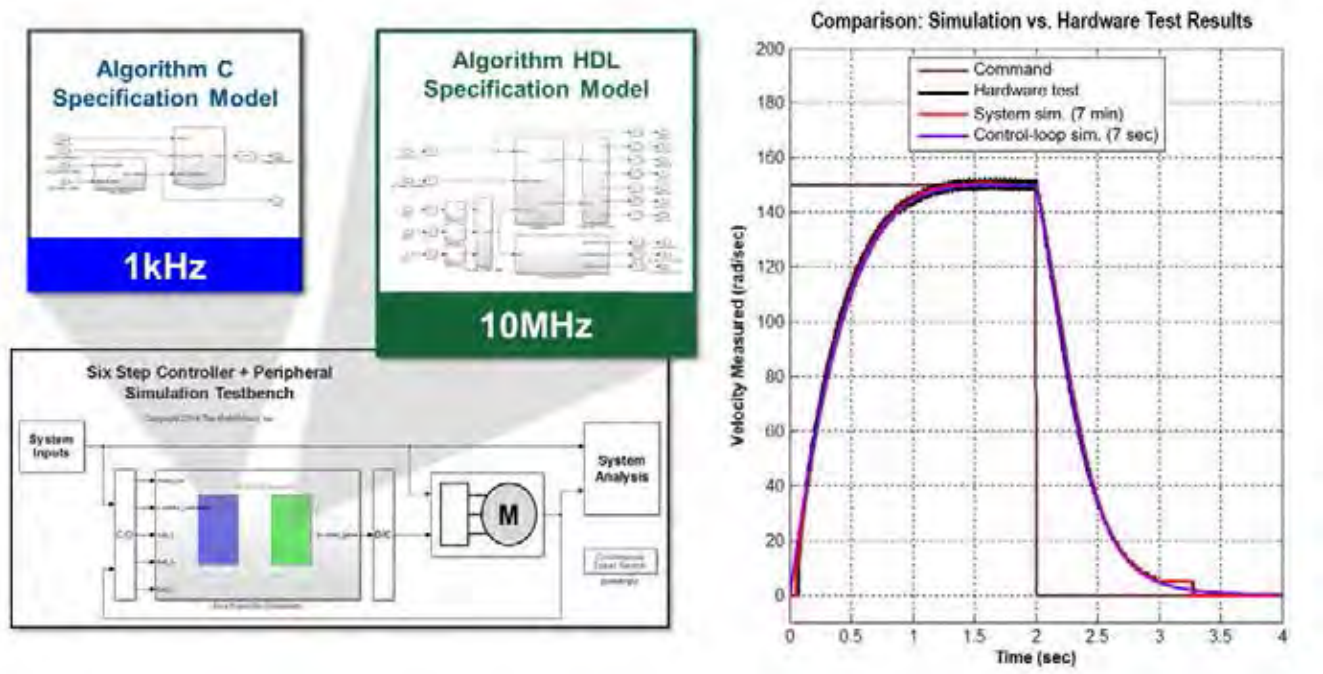


Figure 3 – Hardware and software simulation models used to validate against hardware results

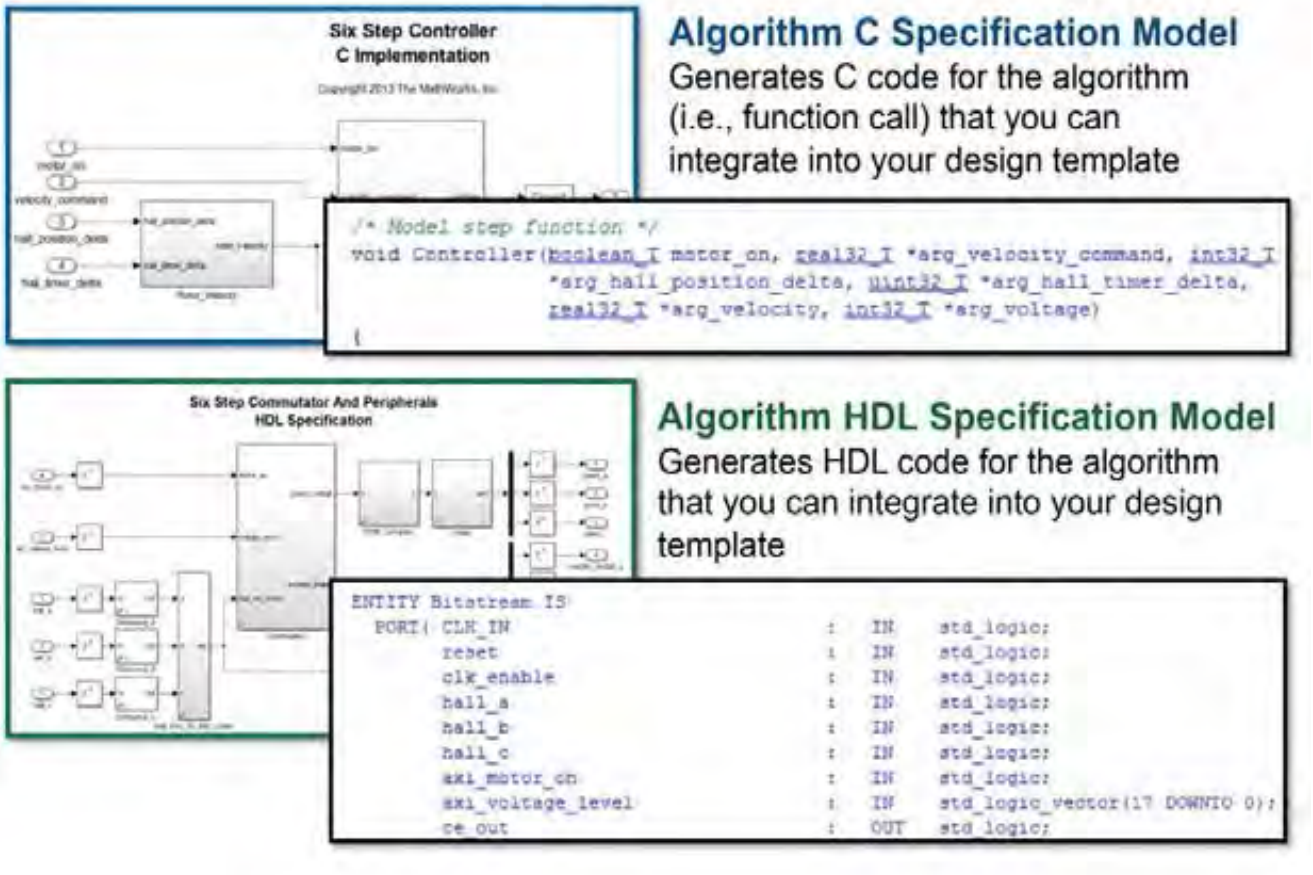


Figure 4 – C and HDL code generated from partitioned Simulink model

With the control-loop gains set, we now can test on a more accurate system model for the controller. In contrast to the control-loop model, the system model incorporates more detailed models of the drive electronics and, more significantly, it includes detailed models that specify the implementation of the controller and peripherals, including timing-accurate models for PWM and Hall-effect sensor processing.

We have partitioned the controller for the Zynq SoC, with the velocity controller and velocity estimator running on an ARM core at 1 kHz and the commutator, Hall sensor and PWM running on the Zynq SoC's programmable logic.

We can compare simulation results for the control-loop and system models (system model results appear as the red signal in Figure 3). In general we get very good agreement between the waveforms, except as the motor's rate approaches zero. At such points the coarseness of the Hall sensor, with only six index pulses per revolution of the motor's shaft, becomes evident. This high-fidelity system model runs the 4-second simulation in 7 minutes, compared with a run-time of only 7 seconds for the lower-fidelity control-loop model. For control system designers, the takeaway here is that these simulation results give us more confidence that the control-loop model is sufficiently accurate for further evaluation of controller alternatives, which can be validated before hardware testing using the system model.

Armed with these findings, we are prepared to prototype the controller on the Intelligent Drives Kit. Through the Zynq SoC guided workflow, we can generate C and HDL code from Simulink models that have been partitioned into subsystems targeting an ARM core and programmable logic (Figure 4).

With this workflow, we use the HDL Coder from MathWorks to generate an IP core that will run in the Zynq SoC device's programmable logic to build an executable running on an ARM core and to establish the interfaces between core and executable over the AXI bus.

With the bitstream loaded into the programmable logic and the executable running on an ARM core, we can run a hardware-in-the-loop test. For this test, we use a modified Simulink testbench model from which we have removed the models for the drive electronics, motor and sensors, since we are using hardware-in-the-loop in place of simulated plant models. To help us check the outcome of the test—and compare it with our simulation results—we can set up the Zynq SoC to store motor shaft velocity measurements and other data in the memory of an ARM core (the black-shaded signal in Figure 3 shows results from hardware testing). Doing so enables us to upload the results to a MATLAB session for processing and visualization at the conclusion of the test by applying a pulse input in the testbench. In this way, we can exactly repeat in hardware the test we've done in simulation. The results from prototyping align very closely with our simulation results, including the discontinuity in the measured motor velocity due to the Hall sensor.

This brief overview illustrates how the MathWorks workflow for the Zynq SoC enables model-based design for use in simulation and prototyping. To continue on into production, you can import the generated C and HDL code into the Vivado® Design Suite, where you can integrate them with executive routines, networking IP and other design components required for the complete system implementation.

To download the models shown in this article and learn more about how to use model-based design with the Zynq-7000 All Programmable SoC / Analog Devices Intelligent Drives Kit from Avnet, visit *mathworks.com/zidk*. From this page, you can also browse a Simulink model that implements a complete field-oriented control model on a Zynq SoC device and view videos showing this example in greater detail.

For information on how MathWorks products support the Xilinx Zynq-7000 All Programmable SoC family, visit *mathworks.com/zynq*.

FPGA
Boards & Modules

EFM-02
FPGA module with USB 3.0 interface. Ideal for Custom Cameras & ImageProcessing.

- Xilinx™ Spartan-6 FPGA XC6SLX45(150)-3FGG484I
- USB 3.0 Superspeed interface Cypress™ FX-3 controller
- On-board memory 2 Gb DDR2 SDRAM 64 Mb Dual SPI flash
- Samtec™ Q-strip connectors 191 (95 differential) user IO

EFM-01
Low-cost FPGA module for general applications.

- Xilinx™ Spartan-3E FPGA XC3S500E-4CPG132C
- USB 2.0 Highspeed interface Cypress™ FX-2 controller
- On-board memory 4 Mb SPI flash
- Standard 0.1" pin header 50 user IO

CESYS
Hardware • Software • HDL-Design
www.cesys.com