Motorcycle Control Prototyping Using an FPGA-Based Embedded Control System

CARROLL DASE, JEANNIE SULLIVAN FALCON, and BRIAN MACCLEERY

To develop a research engine control unit (ECU) that would be suitable for prototyping engine control algorithms as well as sensor and actuator design, we needed reliable, high-performance hardware and custom software. Motorcycles, automobiles, and trucks ship with factory ECUs designed to maximize performance while minimizing cost. However, factory ECUs have closed software and hardware. The control gains, lookup tables, and constants used in the control algorithms are also calibrated for a particular engine design. The development and use of a research ECU allow engine researchers to set up a control system with baseline performance equal to the factory ECU and then investigate methods for improving control performance.

Our project involved the development of a research ECU for the 2004 Yamaha YZF-R6 motorcycle. Engine control requires deterministic loop times on the order of milliseconds as well as precise fuel and spark timing on the order of microseconds. In addition, the target engine revs to 15,500 r/min. At this speed, there is less than 4 ms per crankshaft rotation, and the system must precisely control fuel and spark events in the angle domain to less than 1°.

The chosen approach to this problem depended heavily on field programmable gate arrays (FPGAs). An FPGA is a chip that consists of unconfigured logic gates. Unlike the fixed, vendor-defined functionality of an application-specific integrated circuit (ASIC) chip, an FPGA can be configured and reconfigured for different applications. FPGAs are used in applications where the cost of developing and fabricating an ASIC is prohibitive or the hardware must be reconfigured after being placed into service. FPGAs appear in devices such as electronic instruments, consumer electronics, automobiles, aircraft, copy machines, and application-specific computer hardware.

Because FPGAs can be used to implement custom algorithms in hardware, they offer benefits over processors, such as precise timing and synchronization, rapid decision making, and simultaneous execution of parallel tasks. However, floating-point processors have a computational advantage since FPGAs are limited to integer math. This limitation can be partially overcome by using high-level development tools, code, or IP cores that convert floating- point calculations to integer math. However, a combined FPGA/processor architecture can be used in control applications to take advantage of the strengths of each device.

Drivven, Inc.'s library [1] includes cores for tracking the angular position of crankshafts from a variety of position-sensing formats and generating precise angle-based fuel and spark commands. These tools provide a seamless path from prototype to production for FPGA-based powertrain controllers. Because this path includes early stages of prototyping, where flexibility and computing power are paramount, we often choose PC-based hardware. For this project, which involved engine-control algorithm development as well as road testing, we chose a National Instruments (NI) CompactRIO (Figure 1) embedded control system because of its flexibility, small size, and rugged form factor. With this system, we can add sensors and actuators while readily visualizing the data. In addition, we can mount the controller in the limited space available in a super-sport motorcycle. The unique computational feature of the CompactRIO system is that it includes both a real-time processor and an FPGA. Both devices are programmable using the LabVIEW graphical development environment. With this combined architecture, multiple control approaches and algorithms can be quickly designed and tested on the motorcycle.

Using the CompactRIO embedded control system, we prototyped a new engine control system for the Yamaha

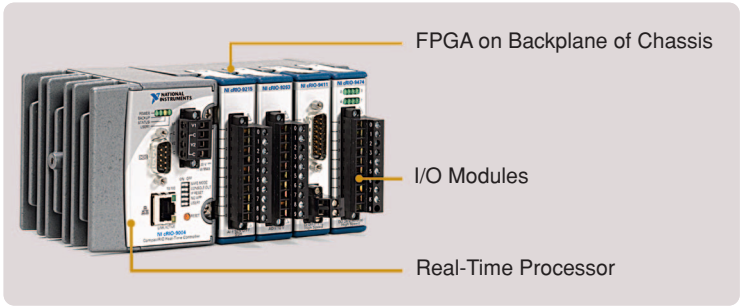

FIGURE 1. CompactRIO embedded control system (4-slot configuration) from National Instruments. The real-time processor in the device controller is connected to an FPGA on the backplane of the chassis through an internal PCI bus. A fully loaded 4-slot CompactRIO system weighs approximately 1.68 kg (3.71 lb) excluding a battery or power supply. The CompactRIO system, without external connectors attached, is 17.8 cm × 8.9 cm × 8.9 cm (7.0 in × 3.5 in × 3.5 in).

1066-033X/06/$20.00©2006IEEE

YZF-R6 motorcycle. CompactRIO completely replaced the original ECU located under the seat. Figure 2 shows the placement of the CompactRIO system in the tail of the motorcycle. The factory ECU was originally located underneath the seat but was removed after initial testing to reverse engineer the software and the calibrations. The CompactRIO system is larger than the factory ECU and thus cannot be placed under the seat. Once an engine control system has been tested with the research ECU, a modified factory ECU can be designed and optimized to meet size and cost constraints.

The modified motorcycle was test driven extensively. Experienced drivers could not detect significant differences between the factory ECU and CompactRIO control system. Thus, the CompactRIO system was shown to be an effective research ECU for the motorcycle. In the future, new control algorithms can be designed and tested using the CompactRIO system.

FIGURE 2. The modified 2004 Yamaha YZF-R6 motorcycle. The factory ECU was removed from under the seat and replaced with the National Instruments CompactRIO embedded control system in the tail of the motorcycle.

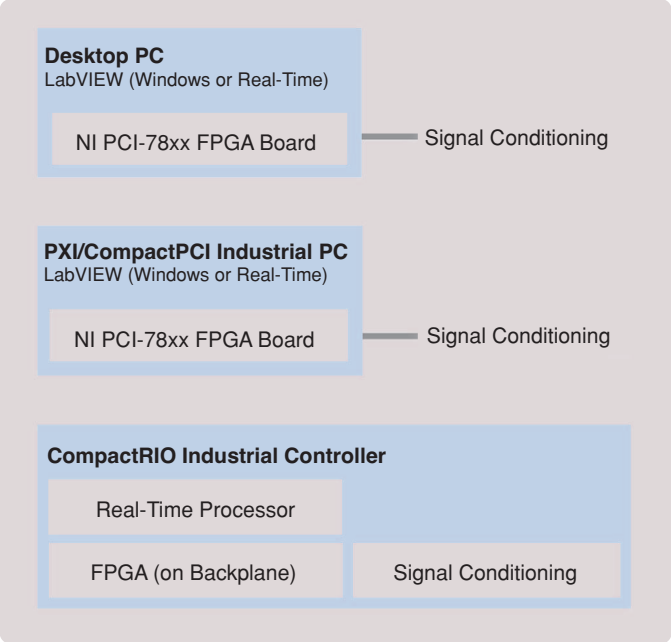

FIGURE 3. Shared hardware architecture diagram for the National Instruments FPGA/processor platforms including CompactRIO, PXI/cPCI, and the desktop PC. All of the platforms can be programmed with LabVIEW Real-Time for the processor and LabVIEW FPGA for the field programmable gate array. The combined FPGA/processor architecture facilitates the design of hierarchical control systems and parallel processing for control, filtering, monitoring, and safety.

COMBINED PROCESSOR AND FPGA ARCHITECTURE

The NI CompactRIO embedded control system is one of several hardware platforms with combined processor and FPGA technology. NI also provides programmable plug-in FPGA boards for the PXI/cPCI Industrial Computer platform as well as for desktop computers.

The shared hardware architecture among the platforms is illustrated in Figure 3. The desktop PC and PXI/cPCI Industrial PC can run Windows or LabVIEW Real-Time [2] on the main floating-point processor. LabVIEW Real-Time allows LabVIEW diagrams to run on top of a commercial real-time operating system for multiple types of targets. In both the desktop and the PXI system, a plug-in FPGA board can be connected to the processor through the PCI bus. The boards are available with a combination of analog and digital I/O.

The CompactRIO architecture is similar to that of the desktop and PXI systems with FPGA plug-in boards. The floating-point processor, programmed with LabVIEW Real-Time, is connected to the FPGA on the backplane through an internal PCI bus. The FPGA is, in turn, connected to the analog and digital I/O modules in a star topology. The CompactRIO modules contain conversion circuitry (for analog modules), isolation, signal conditioning, and breakout enabling direct connection to devices such as relays, solenoids, and motors.

Because of the similar hardware architecture, a common software application architecture can be used for all three platforms. This software architecture shown in Figure 4 includes a host program running on a Windows PC, a real-time program running on the processor, and an FPGA program running in hardware on the FPGA. The processor can run multiple loops at varying priorities including a time-critical loop for both control and time-critical communication with the FPGA. The FPGA can run multiple loops in parallel for control, filtering, monitoring, and safety.

Two traditional methods are available for creating FPGA designs, namely, the text-based programming languages VHDL and Verilog. These languages, known collectively as hardware description languages (HDLs), are

analogous to C for microprocessors. HDLs are powerful but require high levels of expertise to program. Higher level tools are available for simplifying this practice. Examples include the Xilinx System Generator for DSP (includes system modeling and automatic code generation from Simulink and MATLAB) [3], Celoxica's various C compilers [4], and LabVIEW FPGA [5].

For our project, the NI LabVIEW FPGA Development Module was used because of the integration with the CompactRIO tool chain. This development module contains several high-level blocks for control and signal processing that approximate floating-point implementation using the integer math available with the FPGA. In addition, NI makes several toolkits that can be used with LabVIEW FPGA, namely, the Digital Filter Design Toolkit, the SoftMotion Development Module, and the State Diagram Toolkit [6].

All of the low-level FPGA code, or IP cores, are written in Verilog and wrapped in a graphical LabVIEW FPGA HDL Node block, allowing the underlying cores to be used in non-LabVIEW systems such as production engine controllers. Figure 5 shows the high-level LabVIEW FPGA code used in this research.

The combined software and hardware architecture facilitates the development of hierarchical-control strategies. Some lower level control loops can take place completely in the FPGA. Higher level control loops can be written for the floating-point processor in the system. In addition, tasks such as custom startup and shutdown behavior, inline signal processing, as well as custom timing, synchronization, and digital communication, can be implemented in the FPGA. Time-critical parallel tasks such as health monitoring and fault detection can be developed on the FPGA as well.

While processors run computations in sequence, FPGAs can run computations in parallel. Although advanced processor architectures involve multiple cores, the FPGA is still the best choice for parallel computation. If multiple control loops or tasks are implemented in a processor, the overall loop rate decreases. Since multiple control loops and tasks implemented in an FPGA can execute independently, extremely high loop rates can be achieved. The limiting factor for FPGAs is the number of gates and the speed of the I/O hardware.

CUSTOM I/O MODULE DEVELOPMENT

NI has developed a family of analog and digital I/O modules with onboard signal conditioning and breakout connectors for the CompactRIO system. Some applications, however, may require specialized modules that are not

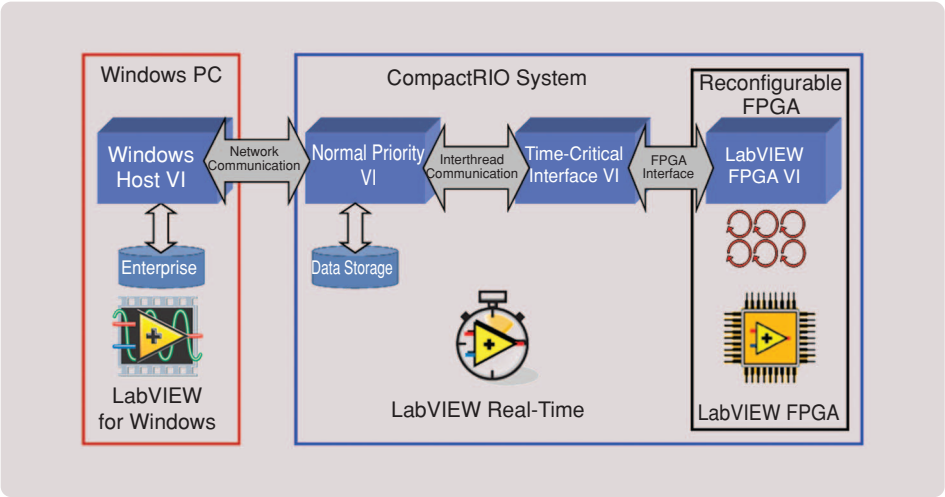

FIGURE 4. Shared software application architecture for National Instruments FPGA/processor platforms. A host PC communicates with the real-time target. The target contains a floating-point processor and an FPGA. The processor can run multiple loops at varying priorities including a time-critical loop for both control and time-critical communication with the FPGA. The FPGA can run multiple loops in parallel for control, filtering, monitoring, and safety.

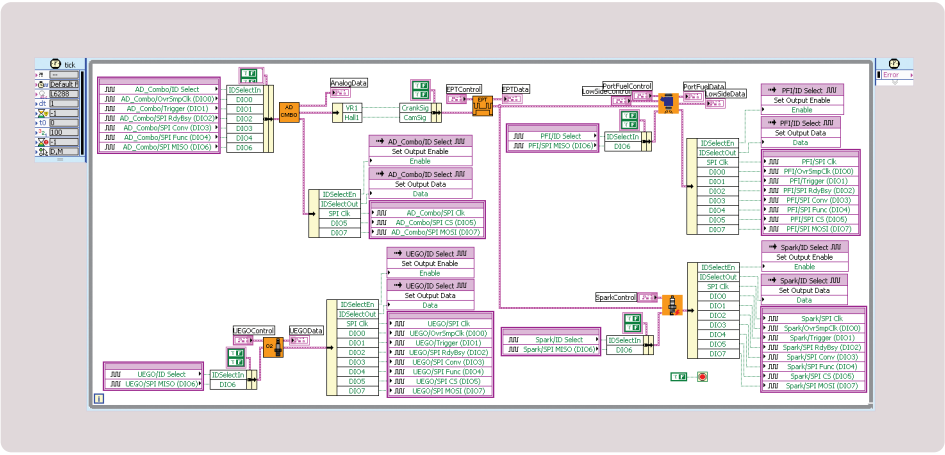

FIGURE 5. High-level LabVIEW FPGA code for motorcycle engine control using National Instruments CompactRIO. The code shows a single-cycle timed loop (similar to a while loop) and hierarchy through subprograms. The single-cycle timed loop ensures that all code within the loop executes in a single 25-ns clock cycle of the FPGA.

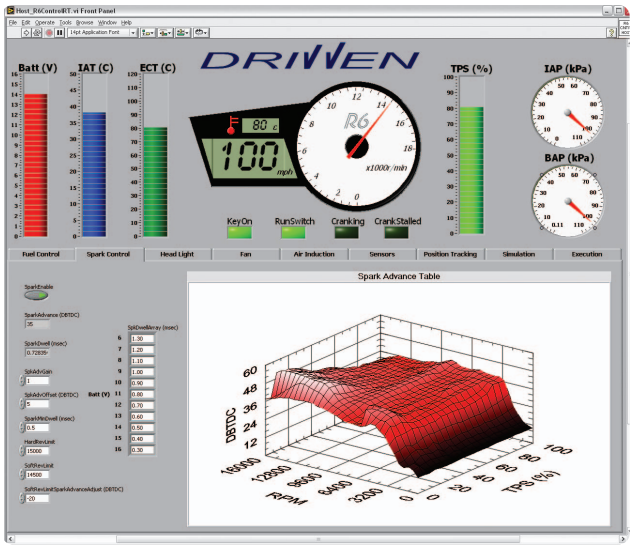

FIGURE 6. LabVIEW host program user interface running on Windows. This program is used to communicate with the CompactRIO system over a wireless Ethernet link.

included in the NI product offering. For these cases, NI provides a CompactRIO module development kit with the software, hardware, and documentation required to create custom modules.

Drivven created three custom CompactRIO I/O modules. The first module, the A/D combo module, provides 22 single-ended, 12-bit analog inputs, two variable reluctance (VR) sensor inputs, and two Hall-effect sensor inputs. This module implements lowpass analog filters as well as over/under voltage protection on all inputs. The second module provides four channels for driving low-impedance port-fuel injectors and four low-side, inductive-load switches for driving general-purpose solenoids. Each channel can be diagnosed for open or short circuits and disabled without CPU intervention. The third module provides eight low-side inductive-style drivers for ignition coils. Each module is designed with low-cost circuitry for prototyping a production-oriented control system. As a result, developers can realize the same I/O behavior in both prototyping and production stages. These three modules monitor all of the motorcycle sensors and control its actuators. We have developed additional CompactRIO modules for powertrain control applications, including modules for an electronic throttle driver, oxygen sensor controller, and common rail diesel injector driver modules. Drivven is working toward providing a complete selection of CompactRIO modules for interfacing with all types of automotive sensors and actuators.

USER INTERFACE AND WIRELESS COMMUNICATION

LabVIEW provides customizable user interface and visualization capability. Figure 6 is a screen shot of the user interface developed for this project. The interface has the ability to adjust all of the calibration values of the engine control code on the fly. The data communicated between the laptop-based interface application and the CompactRIO real-time control application is transported through the Ethernet TCP/IP protocol over a wireless link. A LabVIEW wizard generates LabVIEW code for communicated parameters and tables by TCP/IP. The wireless link proved to be beneficial and can be maintained over short distances (about 300 ft) between a chase vehicle and the motorcycle during road testing. Breakdowns in the wireless link do not affect the operation of the real-time control application.

MAPPING THE FACTORY ECU

The factory ECU was tested and mapped to reverse engineer the software and calibration of the control gains, lookup tables, and constants. Once this was done, baseline engine control performance for the motorcycle was established. In this phase, we carefully tapped into critical motorcycle sensors and actuators using CompactRIO, and logged their signals and events at 200 Hz to the CompactRIO Flash file system. The signals and events include intake air pressure and temperature, barometric air pressure, coolant temperature, throttle position, crank position, cam position, fuel injection start angle and pulse width, and spark advance. Our FPGA-based engine management code was used to track the position of the crankshaft (0.3° resolution) and capture the angle-based timing of the fuel injection and spark events. For this process, the motorcycle was driven along a straight road with minimal traffic, and thus it was not necessary to remove the engine from the motorcycle and install it on a dynamometer.

We recorded the ECU data to 1 Mb files (up to 20 files at 1 file per minute) while riding the bike at numerous combinations of throttle position and engine speed (nearly 700 operating points) to fully map the behavior of the factory ECU. The rider carefully drove the motorcycle so as to reduce transient operation as much as possible. An engineer in a chase vehicle periodically transferred the data files by wireless FTP from the CompactRIO to a laptop and analyzed them for coverage of operating points. A laptop-based NI LabVIEW application sorted the data into speed/load operating tables while filtering out transient data. A mean and standard deviation were calculated from the data for each operating point. In two hours, the team acquired data for 90% of the motorcycle's operating points, which was ample coverage for understanding the mapping of the factory ECU. Later, in the lab, engineers processed the data with LabVIEW, which provided three- and two-dimensional visualization while graphically modifying the raw data to fill in the missing operating points.

ENGINE CONTROL

In the final phase, we used CompactRIO to prototype a research-oriented ECU, achieving performance comparable to the factory ECU while providing the ability to carry

out control algorithm research and development, which is not possible with production-oriented electronics. With CompactRIO, we implemented several of our engine-management FPGA cores, all of which have configurable LabVIEW FPGA icons placed on the block diagram. We can port these same cores directly to production FPGA-based controllers. Using LabVIEW Real-Time, we implemented a combination of speed-density and alpha-N engine control strategies commonly found in high-performance race applications.

A speed-density engine control method monitors the intake-air pressure and temperature to calculate the theoretical mass of air that enters the combustion chamber on each cylinder's cycle. The speed of the engine, however, affects the actual mass of air that enters the chamber due to restrictions and tuning effects of the air-intake and exhaust tracks. Users can characterize this behavior by a one-dimensional lookup table of volumetric efficiency (Ve) values versus engine speed. Users can then calculate a fuel injection mass based on the fuel's stoichiometry (for gasoline, about 14.7 parts air to one part fuel). Many passenger-car engine controllers use the speed-density method for open-loop control until emissions subsystems are operational for closed-loop control. When modifying the intake or exhaust systems, the speed-density control requires that only the Ve table be updated.

In contrast to speed-density control, an alpha-N engine control method looks up the empirical mass of air for each throttle angle (alpha) and engine speed (N) operating point. A two-dimensional lookup table of several hundred points is required. Many high-performance and race engine controllers rely on this method since the intake air pressure does not have enough variability over the entire throttle/load range to effectively use a speed-density method. When users make mechanical modifications to these engines, the operating points must be recalibrated.

We used a combination of these control strategies by applying speed density to low-speed and low-load operating points where intake air pressure had the most variability. An alpha-N method was applied to the rest of the operating map. After we noted Yamaha's use of sensors on the production motorcycle, we determined that the factory ECU was likely implementing a similar strategy. We used the data acquired in the mapping phase to calibrate these control strategies. Experienced riders could not identify significant differences between the factory ECU control and the prototype control. Most importantly, we achieved this level of control without expensive dynamometer time. We successfully achieved our goal of prototyping a motorcycle ECU on time and on budget.

CONCLUSIONS

In past projects, we spent at least two man-years and US$500,000 dollars to develop similar research ECU prototyping systems from custom-designed hardware. For this project, the equipment costs, including the motorcycle and CompactRIO, were US$15,000 and required about three man-months of time. Compact RIO, LabVIEW Real-Time, and LabVIEW FPGA delivered the reliability and precise timing resources required, and the system was rugged enough to withstand the high temperatures and vibrations of the operating environment.

The LabVIEW function blocks (VIs) and application templates for general gasoline and diesel engine control are available through Drivven. Modifying a four-cylinder application to a V-8 application consists of inserting an additional injector driver module and adding the associated interface VI to the FPGA block diagram.

Drivven and NI are jointly sponsoring the Formula SAE team at Colorado State University. Dr. Rudy Stanglmaier is the advisor. The team is using a Yamaha R6 engine, National Instruments CompactRIO, and the work described here.

AUTHOR INFORMATION

Carroll Dase is the president of Drivven, Inc., in San Antonio, Texas. He has designed research and production powertrain control hardware and software for Southwest Research Institute and Motorola. He has also provided powertrain control consulting services to automotive OEMs. He has a B.S. in mechanical engineering from the University of Texas.

Jeannie Sullivan Falcon is a senior engineer in control and simulation at National Instruments in Austin, Texas. She has led research in control and mechatronics at the Air Force Research Laboratory, Midé Technology Corporation, and National Instruments. She received a B.S. in physics from Carnegie Mellon and an M.S. and Ph.D. in mechanical engineering from MIT.

Brian MacCleery is currently the senior product manager for industrial control and embedded design at National Instruments in Austin, Texas. He has a B.S. and M.S. in electrical engineering from the Virginia Polytechnic Institute and State University.

REFERENCES

[1] Drivven, Inc. homepage, "Driven—Powertrain Rapid Control Prototyping," [Online]. Available: http://www.drivven.com

[2] National Instruments LabVIEW Real-Time Web portal, "LabVIEW Real-Time for measurement and control," [Online]. Available: http://www.ni.com/realtime

[3] Xilinx product page, "System Generator for DSP—the leading-edge modeling and implementation tool for high performance DSP systems," [Online]. Available:http://www.xilinx.com/ise/optional_prod/system_generator.htm

[4] Celoxica homepage, "Welcome to Celoxica—the technology leader in C based design and synthesis," [Online] http://www.celoxica.com

[5] National Instruments LabVIEW FPGA Web portal, "LabVIEW FPGA—customize your hardware without having to build it," [Online]. Available: http://www.ni.com/fpga

[6] National Instruments LabVIEW Toolkit Web portal, "LabVIEW Toolkits," [Online]. Available: http://www.ni.com/toolkits