Low-Cost Embedded Controller for Complex Control Systems

Long-Hua Ma, Xiao-Long Shi, Hui Li, Zhe-Ming Lu, and Meng Shao

Abstract—Because of limited resource of embedded platforms, the computational complexity of advanced control algorithms raises significant challenges for the use of embedded systems in complex control field. A Scilab/Scicos based embedded controller is developed on which various control software can be easily modeled, simulated, implemented, and evaluated to meet the ever-expanding requirements of industrial control applications. Built on the Cirrus Logic EP9315 ARM systems-on-chip board, this embedded controller is possible to develop complex embedded control systems that employ advanced control strategies in a rapid and cost-efficient fashion. Due to the free and open source nature of the software packages used, the cost of the embedded controller is minimized.

Index Terms—Complex control, embedded systems, Scilab, system optimization, system-on-chip.

1. Introduction

Recently, with the rapidly-expanding requirements of industrial applications, control systems are becoming more and more complex because of the complex architecture of the system and the computationally time-consuming control algorithms. It is a challenge to develop real-time control systems that are practically applicable. Haber-Haberb et al. [1] proposed a classic solution for the control of a high-performance drilling process. Vitale et al. [2] developed a Matlab-based tool that facilitates a high-level environment in which a new control law can be easily modeled, simulated with real-time constraints and then translated in the appropriate executable format. Centioli et al. [3] utilized the Matlab real time workshop (RTW) for rapid and reliable control implementation in the real time application interface (RTAI) GNU/Linux environment.

With the advances in free and open source software technology for general-purpose computer systems, some researchers attempted to develop control systems based on free and open source software. For instance, in order to evaluate the replacement of the existing expensive commercial solution for the plasma control system, Centioli et al. [4] utilized the open source Linux real-time operating system RTAI on a VME/INTELx86 embedded controller, they concluded that the open source real-time operating system is a good alternative to the commercial product. Mannori et al. [5] presented a complete development chain for industrial control systems using free and open source software, including running controller with real plant and automatic code generation for real time embedded platforms. More recently, Gomez and Mannori [6] concluded that the open source software can be a valid alternative to closed source solution, not only for economic reasons but also for intrinsic flexibility of the development model that matches most of the technical challenges of the embedded world.

The rest of this paper is structured as follows. In Sections 2 to 4, we describe the design of the hardware and software components of the embedded controller of the complex control systems. Section 5 gives some essential implementation details. In Section 6, we construct a virtual laboratory based on the developed embedded controller and conduct experiments to test the performance of the system. We conclude the paper in Section 7.

2. Complex Control Systems

In this paper, we develop an embedded controller for
complex control applications. The key software used is the Scilab/Scicos package\(^{[12]}\), a free and open source alternative to commercial packages for dynamical system modeling and simulation such as Matlab/Simulink. Since hardware devices are becoming cheaper by the day, the software development cost has dominated the cost of most embedded systems. As a consequence, the use of the free and open source software facilitates to minimize the cost of the embedded controller. This implies that the developed embedded controller is extremely low-cost thanks to the zero cost of the software packages used.

On the other hand, Scilab is a software package providing a powerful open computing environment for engineering and scientific applications. It features a variety of powerful primitives for numerical computations. There exist a number of mature Scilab toolboxes, such as Scicos, fuzzy logic control, genetic algorithm, artificial neural network, model predictive control, etc. All these features of Scilab make it possible and quite easy to implement complex control algorithms on the embedded platform we developed in this work.

To satisfy the ever-increasing requirement of complex control systems with respect to computational capability, we use the Cirrus Logic EP9315 ARM chip in this project. This chip has a Maverick Crunch math coprocessor, which can accelerate the computing speed 10 even 100 times faster than the pure ARM920T core. The platform runs on an ARM-Linux system. Since Scilab and Scicos were originally developed for general-purpose computers such as desktop PCs, we port Scilab/Scicos to the ARM-Linux platform\(^{[9]}\). Several interfaces and toolboxes are implemented to facilitate embedded control.

With the developed platform, the design and implementation of a complex control system will become relatively simple, as shown in Fig. 1. The main procedures involved in this process are as follows: 1) model, design, and simulate the control system with Scilab/Scicos on a host PC and 2) download the well-designed control algorithms to the target embedded system. The Scilab code on the embedded platform is completely compatible with that on the PC. Consequently, the development time can be significantly reduced.

3. Hardware Platform

3.1 SoC System

In recent years, SoC has become one of the focal points in modern science and technology development around the world. It is believed to be more cost effective than a system in package. One of the most typical application areas of SoC is the embedded system.

In this work, the processor of SoC is the Cirrus Logic EP9315 ARM9 chip\(^{[13]}\). A snapshot of the hardware board is shown in Fig. 2. The main features of the hardware include:

- Support Linux operating system
- 64 MB of SDRAM
- 32 MB of flash memory
- A/D (analogue-to-digital), D/A (digital-to-analogue) Interfaces
- Three UARTs (DB9 Connector)
- Two full-speed USB host connections
- 10/100 Mbps Ethernet
- Support for analog video graphics array (VGA) and liquid crystal display (LCD) connections
- Four-wire touch screen interface
- MaverickCrunch coprocessor

Using this SoC board, it is easy to communicate with other components of the system, for example, sampling data from sensors and sending control commands to actuators. This SoC board also support A/D, D/A, Serial and Ethernet interfaces. To keep the system user-friendly, the embedded controller includes a LCD with touch screen.

3.2 MaverickCrunch Coprocessor

The MaverickCrunch coprocessor\(^{[11]}\) is an advanced, mixed-mode math coprocessor that greatly accelerates the
single and double-precision integer and floating-point processing capabilities of the ARM920T processor core. The MaverickCrunch coprocessor uses the standard ARM920T coprocessor interface and shares its memory interface and instruction stream. The coprocessor operates in parallel with the main processor. The processor receives instructions from a single 32-bit instruction stream.

The MaverickCrunch coprocessor accelerates IEEE-754 floating point arithmetic and 32-bit fixed point arithmetic operations such as addition, subtraction, multiplication, etc. It provides an integer multiply-accumulate (MAC) that is considerably faster than the native MAC implementation in the ARM920T. The single-cycle integer MAC instructed in the MaverickCrunch coprocessor allows the EP9315 to offer unique speed and performance while dealing with math-intensive computing and data-processing functions in industrial electronics.

Table 1 lists the time needed to execute every test function 360000 times. Compared with the case without the MaverickCrunch coprocessor, the computational speed of the system becomes 10 to 100 times faster when the MaverickCrunch coprocessor is used. The powerful computational capability makes the EP9315 platform suitable for complex control applications.

### 4. Software Design

There are a number of considerations in implementing control algorithms on embedded platforms including the ARM9 board we use. One of the most important things is that embedded platforms are usually limited in resource such as processor speed and memory. Therefore, control software must be designed in a resource-efficient fashion.

<table>
<thead>
<tr>
<th>Functions</th>
<th>ADD</th>
<th>SUB</th>
<th>MUL</th>
<th>SIN</th>
<th>LOG</th>
<th>EXP</th>
</tr>
</thead>
<tbody>
<tr>
<td>HPF (ms)</td>
<td>1</td>
<td>1</td>
<td>25</td>
<td>950</td>
<td>950</td>
<td>902</td>
</tr>
<tr>
<td>SFP (ms)</td>
<td>187</td>
<td>190</td>
<td>310</td>
<td>7155</td>
<td>7468</td>
<td>6879</td>
</tr>
<tr>
<td>Ratio</td>
<td>1:187</td>
<td>1:190</td>
<td>1:12.8</td>
<td>1:7.6</td>
<td>1:7.8</td>
<td>1:7.6</td>
</tr>
</tbody>
</table>

*HFP: Hardware float point, with the MaverickCrunch coprocessor
*SFP: Software float point, without the MaverickCrunch coprocessor

The key software packages used in this paper include Linux, TinyX, JWM, Scilab/Scicos, Scilab supervisory control and data acquisition (SCADA) toolbox (the SCADA stands for supervisory control and data acquisition), and other related Scilab toolboxes. The system software architecture is shown in Fig. 3.

#### 4.1 Software Packages

1) Linux. The developed embedded controller is built on the Linux kernel. The Linux kernel provides a level of flexibility and reliability simply impossible to achieve with any other operating system (OS) such as Windows, Unix and Mac OS. Mainly for this reason, many embedded systems choose Linux OS. Recent Linux kernels (2.6.x) are able to offer almost hard real-time performance for sampling times up to one millisecond, which is adequate for control applications in general.

2) TinyX. TinyX is an X server written by Keith Packard. It was designed for low memory environments. On Linux/x86, a TinyX server with RENDER support but without support for scalable fonts compiles into less than 700 KB of text. TinyX tends to avoid large memory allocations at runtime and tries to perform operations on-the-fly whenever possible. Unlike the usual XFree86 server, a TinyX server is completely self-contained. A TinyX server does not require any configuration files, and will function even if no on-disk font is available. All configurations are done at compile time and through command-line flags. It is easy to build user-specified graphical user interface (GUI) applications with TinyX. More information about TinyX can be found at http://www.xfree86.org.

3) Scilab. Scilab/Scicos is utilized in this work to build the development environment for control software executing control algorithms. Fig. 4 shows some GUIs of the Scilab environment. Developed initially by researchers from Institut national de recherche en informatique et en automatique (INRIA) and Ecole nationale des ponts et chaussées (ENPC), France, since 1990, Scilab is currently a free and open source scientific software package for numerical computations. Scilab has many toolboxes for modeling, designing, simulating, implementing, and evaluating hybrid control systems. It is now used in academic, educational, and industrial environments around the world. Scilab includes hundreds of mathematical functions with the possibility to add interactively programs from various languages, e.g., Fortran, C, C++, and Java. It has sophisticated data structures including lists, polynomials, rational functions, and linear systems. Scilab is an interpreter and a high level programming language.

4) Scicos. Although it is possible to model and design a hybrid dynamical system by writing scripts using the primitives of the Scilab language, this is often time consuming and the developers are prone to insert bugs during the manual coding. To simplify this task, Scilab includes a graphical dynamical system modeler and simulator toolbox called Scicos. Scicos can be used in...
control, communication, signal processing, queuing systems, studying of physical, and biological systems, etc. Using the Scicos graphical editor, it is possible to model and simulate hybrid dynamic systems by simply placing, configuring and connecting blocks. To achieve easy customization, maximum flexibility and complete integration with Scilab, most of the Scicos GUIs are written in the Scilab language.

5) Scilab SCADA toolbox. To facilitate data acquisition and control operations, we develop the Scilab SCADA toolbox that the interfaces Scilab with several kinds of I/O ports including serial port, Ethernet and Modbus on the embedded Linux system. These communication interfaces make it possible to connect the embedded controller with other entities in the system, e.g. sensors, actuators, and the controlled physical process, using various communication mechanisms or networks.

In a complex and large-scale control system in industry, a huge amount of data, e.g. system output samples and control commands, will be produced during run time. These data usually has to be stored in order to provide support, e.g. historical data query and higher-layer system optimization. To meet this requirement, we developed an interface to MySQL database in the Scilab SCADA toolbox.

In addition, to provide a standard-compatible solution for the industrial control field, the Scilab SCADA toolbox conforms to the OPC (object linking and embedding (OLE) for process control) standard. OPC is a widely accepted industrial communication standard that enables the exchange of data between multi-vendor devices and control applications. It helps provide solutions that are truly open, which in turn gives users more choices in their control applications. The interoperability between heterogeneous entities is assured through the support for non-proprietary specifications. A GUI of the OPC toolbox we develop is shown in Fig. 5.

With this OPC interface, it is possible to use Scilab as the core control software. And the communications with other (third-party) hardware devices and software tools will be effortless. These help to fully exploit the powerful functionalities of Scilab in complex control applications.

4.2 Building Cross-Compilation Toolchain

A cross compiler is a compiler that is able to create executable code for a platform other than the one on which it is run. The basic role of a cross compiler is to separate the build environment from the target environment. It is particularly useful for the development of the embedded controller based on Scilab/Scicos, which typically works in a general-purpose computing environment other than the embedded platform.

To port related software packages from PC to the ARM-Linux system, it is essential to build the cross-compilation toolchain environment at first. There exist several approaches to set up a cross-compilation toolchain. In this work, we build the cross compiler for the ARM-Linux system using the buildroot toolkits. Buildroot is a set of make-files and patches that allow generating a cross-compilation toolchain and a root file for the target system easily. The cross-compilation toolchain makes use of uClibc, which is a tiny C standard library. Several tools, such as bison, flex, and build-essential are also exploited. It is worth mentioning that the g77 compiler option should be enabled during this process. Since most of the Scilab code is written in Fortran, the g77 compiler is necessary when compiling Scilab.

4.3 Porting Scilab/Scicos to ARM-Linux

The Scilab/Scicos package was originally designed for general-purpose computer systems, e.g. desktop PC. In order to develop Scilab-based embedded controller, it is necessary to port this software package to the ARM-Linux platform. There are several considerations to take into account when addressing this issue. For instance, the GUI system of Scilab/Scicos is based on X11. Consequently, the embedded system must include the X11 server. The GUI should also be cross-complied before Scilab is cross-compiled. To make the handling of Scilab/Scicos more comfortable, a window manager is required. In this work, we take advantage of a very tiny and pure C source window manager, i.e. Joe&apos;s window manager (JWM). In addition, Tcl/Tk can be employed to enhance the
We have successfully ported Scilab/Scicos to the ARM-Linux system. The drivers for several types of communication interfaces have been implemented in the Scilab SCADA toolbox, which is described in Section 3.1. The main steps of this process are as follows:

1) Port Linux to the ARM platform;
2) Port TinyX to ARM-Linux;
3) Port JWM to ARM-Linux;
4) Port Scilab/Scicos to ARM-Linux;
5) Configure and optimize the embedded Scilab/Scicos.

5. Implementation

Based on the above hardware and software design, the embedded controller has been implemented. Fig. 6 gives a snapshot of the embedded controller we develop.

With this embedded controller, the procedures of designing and implementing an embedded control system can be outlined as follows:

1) Specify system requirements;
2) Build the model of the controlled physical processes with Scilab/Scicos;
3) Design control algorithms;
4) Conduct simulations, analyze the results, and adjust related control parameters accordingly;
5) Download control programs to the embedded controller;
6) Test the embedded controller;
7) Deploy the controller in real systems.

The first four steps can be performed on a PC. The powerful capability of such a platform facilitates further improvement of the development efficiency. Since almost all the codes of Scilab/Scicos are platform-independent, the programs and toolboxes will be executable on the embedded controller as long as they can run on the PC.

5.1 Rapid Prototyping of Control Algorithms

Modern control design methods allow creating complex multivariable control laws. The need for a transparent and straightforward design process often leads to the software implementation of controllers. The state-space approach is a unified method for modeling and analyzing nonlinear and time-invariant systems. Generally, the mathematical equations can be divided into two parts: a set of equations relating the state variables \( x \) to the control input \( u \), and a second set of equations relating the state variables and the current input to the system output \( y \). The general form of the state-space equations modeling control systems is

\[
\frac{d}{dt} x = Ax + Bu \\
y = Cx + Dy
\]

where \( A, B, C, \) and \( D \) are matrices in appropriate dimensions. This model can be easily implemented in Scilab since it supports all basic operations on matrices such as addition, multiplication, concatenation, extraction, and transpose, etc.

The use of Scilab makes it easy to model, design, and implement complex control algorithms in the embedded controller developed in this work. The Scilab has a variety of powerful primitives for programming control applications. Additionally, there are several different ways to realize a control algorithm in the Scilab/Scicos environment. For instance, Scilab can be programmed as a .sci file written in the Scilab language or visualized as a Scicos block linked to a specific function/program written in Fortran or C. In addition, there are an increasing number of toolboxes of Scilab that provide support for implementing advanced control strategies using, for example, the fuzzy logic, genetic algorithm, neural networks, and online optimization.

As an example for system modeling and simulation in Scicos, Fig. 7 shows a control system for a water tank. The models of the controller and the water tank are highlighted by the dashed and solid rectangles. The step response of the control system is plotted in Fig. 8.
5.2 Computational Capability Analysis

Computational capability is a critical attribute of the embedded controller since the execution of the control program affects the temporal behaviour of the control system, especially when complex control algorithms are employed. Therefore, we assessed the computational capability of the developed embedded controller in comparison with that of a PC. The time for executing different algorithms is summarized in Table 2.

6. Experiments Platform

In this section, we test the performance of the developed embedded controller. However, it is costly to build the real controlled physical processes for experiments on complex control applications. For this reason, we constructed a virtual control laboratory to facilitate the experiments on the embedded controller.

6.1 Virtual Control Platform

The schematic diagram of the structure of the experimental system is shown in Fig. 9. The basic idea of the virtual laboratory is utilizing a PC to run a dynamic system modelling software to simulate the physical process to be controlled. The control algorithms are implemented on the embedded controller, which exchanges data with the PC via a certain communication protocol, e.g. serial, Ethernet or Modbus.

Both of the PC and the embedded controller use Scilab/Scicos as core software. Given that they can be modeled using Scilab/Scicos, experiments on various virtual physical processes are possible by this virtual control platform.

Table 2: Comparison of computational capability of PC and ARM

<table>
<thead>
<tr>
<th>Rand (800, 800)</th>
<th>Deloy algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td>PC (s)</td>
<td>0.029</td>
</tr>
<tr>
<td>ARM (s)</td>
<td>1.176</td>
</tr>
<tr>
<td>Ratio</td>
<td>1.40</td>
</tr>
</tbody>
</table>

6.2 Case Study

In the following, the control of a water task is taken as an example for the experimental study. The water tank is modeled as shown in Fig. 10 and implemented on the PC as shown in Fig. 9. The controller implemented on the embedded controller is shown in Fig. 11. The control objective is to keep the water level (denoted by \( y \)) in the tank to 10. The PC and the embedded controller are connected using the Ethernet and they communicate based on the user datagram protocol (UDP).

Fig. 12 depicts the water level in the tank when different sampling periods are used, i.e. \( h = 0.1 \) s, \( 0.2 \) s, and \( 0.5 \) s, respectively. It can be seen that the control system can achieve satisfactory performance. The water level is successfully controlled at the desired value in all cases.

![Fig. 9. Experimental system.](image)

![Fig. 10. Water tank model.](image)

![Fig. 11. Embedded controller.](image)

![Fig. 12. Control performance: (a) \( h = 0.1 \) s, (b) \( h = 0.2 \) s, and (c) \( h = 0.5 \) s.](image)
7. Conclusions

In the work, we developed an embedded controller using Scilab/Scicos. The main features of the developed embedded controller include: 1) the cost is very low thanks to the use of free and open source software packages; 2) it makes possible to implement complex control strategies on the embedded platform through taking the advantage of the powerful computational capacity of Scilab; 3) it helps to reduce the development time of control systems; 4) it is applicable to a variety of fields such as industrial control, system optimization, instrument, and education.

References


Long-Hua Ma was born in Zhejiang, China in 1965. He received his Ph.D. degree in system engineering from the Department of Control Science and Engineering, Zhejiang University in 2001. He is currently working as an associate professor with Zhejiang University. His research interests include navigation systems, information fusion, embedded computing, etc.

Xiao-Long Shi was born in Zhejiang, China in 1989. He is currently studying as a graduate student majoring in navigation guidance and control with Zhejiang University. His research interests include GPS and integrated navigation.

Hui Li was born in Heilongjiang, China in 1977. He received his Ph.D. degree in communication and information system from the Department of Electronics and Information Engineering, Harbin Institute of Technology in 2006. He is currently working as an associate professor with Zhejiang University. His researches include space communication, wireless networks, routing algorithms and coding, etc.

Zhe-Ming Lu was born in Zhejiang, China in 1974. He received his Ph.D. degree in instrument science and technology from the Department of Electric Engineering and Automatics, Harbin Institute of Technology in 2001. He is currently working as a full professor with Zhejiang University. His researches include multimedia single analysis and processing, information hiding, astronautics signal processing, etc.

Meng Shao was born in Zhejiang, China in 1968. She received her B.S. degree in computer application from the Open University of China in 2002. She is currently working as a senior engineer with the Hangzhou First people’s hospital. Her researches include embedded systems, cyber physical systems, etc.