Accelerating PID controller development with rapid prototyping and model-based design

Yue Liu a, *, Yanping Wang a, Zhiping Yan b

a School of Information Engineering, Beijing Institute of Fashion Technology, Beijing, China

b China Textile Machinery & Electric Research Institute, Institute

ARTICLE INFO

Article history:
Available online 22 March 2016

Keywords:
CCS
Code generation
Embedded system
Model-based development
Matlab

ABSTRACT

The development of digital PID controllers for adjustable boost DC–DC converters is discussed in this paper. The primary advantages of using digital signal processors over analogue circuits are higher immunity to environmental changes (such as temperature and ageing), increased flexibility by changing the software, more advanced control techniques, and a reduced number of components. Many engineers, scientists, and researchers rely only on the efficiency of MATLAB language to explore designs, but for system implementation, they were forced to translate high-level MATLAB algorithms into the lower-level language of C. Therefore, a type of advanced method of DSP software development is introduced. The Real-Time Workshop toolbox enables exploration and implementation to be brought together in the MATLAB platform. MATLAB is a technical computing language that can automatically generate readable, compact, and fast C code directly from its algorithms, which could be compiled using the Texas Instruments (TI) Code Composer Studio (CCS) integrated development environment (IDE). With this capability, developers can maintain a single design source and use one language in one development environment from concept to implementation. This model-based development method has a unique value for rapid prototyping.

Copyright © 2015, Far Eastern Federal University, Kangnam University, Dalian University of Technology, Kokushikan University. Production and hosting by Elsevier B.V. This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/4.0/).

Introduction

With the expansion of the applications over a wide range, the complexity of embedded software systems continues to increase; as a result, the development and maintenance has become an essential and complex work. This work represents a very high request for developers. During the development of embedded software, determining how to integrate the newest algorithm and request for developers. During the development of embedded software, determining how to integrate the newest algorithm and control strategy quickly enough becomes one of the focuses of attention of scholars both at home and abroad.

In software design, online debugging, real-time adjustment, and easy maintenance all conform to the needs of product development; in particular, shortening the development cycle has become the focus of the industry. Shortening the product development cycle, improving the reusability of components and enhancing scalability of the control system are the most important issues in product development and design.

Currently, to achieve rapid product development, unified modelling languages are being increasingly applied in the development of embedded systems. Many well-known international companies have launched their own design tool. For example: telelogic launched the UML tool known as Tau Developer, Mirabilis launched the System C modeller, and Mathworks added the Simulink toolbox to support modelling-based embedded system simulation and design. The Simulink tool has its own advantage in system modelling simulation verification and debugging. The Simulink tool can automatically generate readable, compact, and fast C code that can be compiled using the Texas Instruments (TI) Code Composer Studio (CCS) integrated development environment (IDE); these methods have effectively resolved the problem of production efficiency.

Traditionally, during the programming of large-scale applications, the modular approach is required. In a large-scale system, every module depends on each other; this interdependence leads to a problem, i.e., once errors occur during execution, developers must find a bug among hundreds of lines hidden in the source code. This interdependence problem delays the development progress and increases the costs substantially. At the same time, the readability and portability of the code are restricted by hand coding. All
of these disadvantages cause enormous problems during maintenance.

This paper presents a method that integrates development, testing, debugging, and code generation in the software, combined with the DC Boost converter to verify this process.

**Development methods**

This section describes a new method to close the gap between concept and implementation, followed by the presentation of the use of a classical closed-loop DC Boost converter as an example to describe the model-based design in detail. The experimental results indicate that the method will simultaneously meet the requirements of product development; the entire software development cycle is less than a week using the proposed method.

**Model establishment and simulation**

Simulink is one of the most commonly used toolboxes in the Matlab platform. In Simulink, the simulation model is composed of one or more blocks. Each block has its own internal data stream – input data and output data; each block may also have one or more trigger events, which can be set to activate the module. These simple logic modules are the foundation to build the system. All of the models are composed of the corresponding blocks stacked together. They are interdependent and related to each other in a manner to form the whole system. In this way, Simulink can be used for modelling systems. In particular, the embedded control system is based on modern control theory.

**Creating the prototyping environment**

To understand how the control system behaves, we must build a simulation model similar to the algorithm by using a variety of blocks in Simulink. Most of the prototyping systems, such as CCS, only support ECU software development; however, a control system design may also include other code and continuous electronic components to provide feedback to the algorithm. With MATLAB’s Model-Based Design, we can add target microprocessors and FPGAs during the development of the specifications and use of the electronic components to simulate the prototyping environment to verify extend or enhance algorithms.

**From control requirements to algorithm design**

The preparation stage of MATLAB’s Model-Based Design process consists of three main steps: requirements analysis, algorithm design, and testing. In the control requirements analysis phase, we study the target and define the system requirements based on control theory and actual need. Next, we create an initial model of the design in Simulink and use the Control System Toolbox™ to optimise the PID controller and estimate the step response. After the verification, Simulink can estimate the feasibility of the control system. By modifying the parameter, the characteristics of the platform can be rapidly tested in the security virtual environment. This method avoids the problems due to blind parameter selection.

**Automatic code generation and verification**

**Code generation and on-line debugging**

To generate the control algorithm code based on the simulation system, the control block should be peeled apart from the main circuit and saved as the corresponding subsystem file, i.e., the configured input and output interface in Simulink. Finally, we use the Real-Time Workshop (RTW) provided by Matlab to generate the project code for the TI Code Composer Studio. After the compilation, the CCS will link the target processor downloaded embedded code to the target board to complete the development. This paper uses TI 28335 as the main controller. We choose a typical closed-loop control system as an example to demonstrate how to build a platform from concept to a fully functioning system. The diagram of a closed-loop control system model is shown in Fig. 1. The ADC model is used as a reference compared to the given value, with the error becoming the input source of the PID. After the calculation, the algorithm will use the I/O to output the control signal to achieve the goal.

After tuning the parameter simulation, the simulation results demonstrate how the system behaves during disturbances (Fig. 2). If the result satisfied the control requirements, then we could follow the next step of downloading the algorithm into the MCU. For the convenience of online debugging, we chose the RAM download mode for this prototype. First, select the target preferences to configure the storage space in 28335, including the code and data address, in the RAM mapping. Finally, in the real-time workshop interface, click the generate button to have Matlab automatically generate the C code project file for the CCS development environment; have Matlab compile the C code into assembly language code that can be used in 28335. In the process, the following message will appear after the compilation process is completed.

```
### Connecting to Code Composer Studio(tm)
### Generating code into build folder: C:\Boost\DSP_boost_online_ticcs
### Invoking Target Language Compiler on DSP_boost_online.rtw
### Loading TLC function libraries
### Building project...
### Build done.
### Download done.
```

![Fig. 1. Control system diagram.](image)

![Fig. 2. Output signal during a disturbance.](image)
So far, Matlab has established a link to the CCS environment, compiled the algorithm, and loaded the output file. From now on, the algorithm has been completely transferred from a model into actual hardware implementation without the requirement to manually write any code. This method allows the developer to focus on the algorithm design, thereby minimising coding errors.

The last step is tuning the parameters on the actual system based on the behaviour summary from the simulation result. As shown in Fig. 3, it is convenient to optimise the control system's parameters, such as PID, target voltage, and so on, which are shown at the lower right corner of the window. This interface ensures the coding to be error free; in the meantime, this method enables the development efficient to be easy to maintain. This process will have great impact on production efficiency.

**Code generation and on-line debugging**

The experimental results are shown in Fig. 4. Signal 1 is the output signal 1, and Signal 2 is the output value of the PI regulator. In this figure, when in steady-state, the system output value is perfect at 100, as determined by the set target value. When the system encounters an interference pulse, the output decreases and the PI regulator's value increases rapidly to force the output to increase and approach the target value. The actual hardware system response matches the simulation results. This result fully verifies the feasibility and correctness of the developed method described above.

**Conclusion**

This paper briefly described how to use the auto code generation technique with real-time debugging function in the Matlab environment and presented the results of the design of a closed-loop control system on F28335 target board using the proposed technique. Because Simulink generates the executable code corresponding to the simulation, there is no need to write any code by hand through the entire design process. This new method eliminates the defects existing in the traditional embedded code design, which significantly shortens the development cycle and improves the production efficiency.

**Acknowledgements**

This research is completed under the financial aid of the National Science-Technology Support Plan Projects “2013BAH41F00”.

**Further reading**