Chapter 8

On Model-Driven Engineering of Reconfigurable Digital Control Hardware Systems

Tomás Balderas-Contreras
National Institute for Astrophysics, Optics and Electronics, Mexico

Gustavo Rodríguez-Gómez
National Institute for Astrophysics, Optics and Electronics, Mexico

René Cumplido
National Institute for Astrophysics, Optics and Electronics, Mexico

ABSTRACT

The continuous increase in the degree of design complexity during the development process of modern digital hardware systems, come into being due to the increasing demand of more and more functionality under strict design constraints, has led to designers trying to alleviate this complexity by increasing the level of abstraction when describing the functionality of a system. Recent proposals in the field of Electronic Design Automation intend to use common programming languages, like C, C++, and Java, or dialects derived from them, to describe the behavior of a digital hardware system and then generate a lower-level representation, closer to the hardware implementation platform, from such description. This phenomenon led us to firmly believe that the process of describing the functionality of a digital circuit resembles more and more the process of developing software; and, thus, it is possible to experiment with the application of the latest trends in software engineering, like the Model-Driven Engineering (MDE) paradigm, to design digital hardware systems. In this chapter we describe the basic principles of MDE, and provide some hints about the kind of languages and transformation tools needed to design algorithms in the domain of digital control that could be transformed into a digital circuit. We intend to open doors and encourage the research on the design of digital control systems at higher levels of abstraction and their implementations in different kinds of hardware platforms, including reconfigurable devices.

DOI: 10.4018/978-1-60960-086-0.ch008
INTRODUCTION

A digital control system is made up of a set of computer-based components whose mission is to coordinate, manage, and command the operation of another system. Examples of controlled systems include navigation systems for both terrestrial and aerial vehicles, specific-purpose industrial machinery, automated chemical and thermal processes, and other man-made artifacts and dynamic processes. The algorithms that perform these control operations are usually implemented in software running on special microprocessors known as microcontrollers, which are present in plenty of devices, vehicles, and machines that are common in our everyday life.

Figure 1 illustrates the organization of a generic closed-loop digital control system. It is built around a digital computer that receives, as input, a discrete signal $e(kT)$ that is the difference between the sampled versions of both the system’s input ($r(t)$) and the controlled system’s (the plant) output ($y(t)$). The computer produces another discrete signal ($u(kT)$) that, by means of a D/A converter and a zero-order hold (ZOH) module, is transformed into a continuous signal that controls the plant. The sample period is denoted as $T$ in the figure.

According to (Wills, Heck, Prasad, Schrage, & Vachtsevanos, 2001), there are a number of capabilities demanded by modern software used by both civilian and military control systems. These capabilities are driven by modern technological trends and include the following:

- **Adaptability:** Depending on changes in the environment, changes in external inputs and/or the behavior of the system, the control software modifies its own behavior in a quick and seamless manner and without affecting the system’s stability.
- **Extensibility:** The ability to include software components for new technologies (like control algorithms, sensors, or communication protocols) without redesigning the architecture of the control system software.
- **Interoperability:** Current control systems are distributed among a number of heterogeneous hardware/software platforms. These systems should be able to communicate with each other using different protocols and communication links, and meet a number of strict constraints related to response time and reliability.

This sort of software reconfigurability demands that the software architecture of the control system is flexible enough to support on-line changes without compromising the integrity of the system. There are a number of software technologies that might be useful to implement the previous capabilities, like component-based development (Kozaczynski & Booch, 1998) and self-adaptive software (Oreizy et al., 1999).

Alternatively, the control algorithms and operations performed by software may be implemented by a digital hardware system when strict performance and real-time requirements are a
Related Content

RuCAS: Rule-Based Framework for Managing Context-Aware Services with Distributed Web Services
[www.igi-global.com/article/rucas/126616?camid=4v1a](www.igi-global.com/article/rucas/126616?camid=4v1a)

Predicting OSS Development Success: A Data Mining Approach
[www.igi-global.com/article/predicting-oss-development-success/58644?camid=4v1a](www.igi-global.com/article/predicting-oss-development-success/58644?camid=4v1a)

On the Application of Automated Software Testing Techniques to the Development and Maintenance of Speech Recognition Systems
[www.igi-global.com/chapter/application-automated-software-testing-techniques/62149?camid=4v1a](www.igi-global.com/chapter/application-automated-software-testing-techniques/62149?camid=4v1a)

Using Attack Graphs to Analyze Social Engineering Threats
[www.igi-global.com/article/using-attack-graphs-to-analyze-social-engineering-threats/136466?camid=4v1a](www.igi-global.com/article/using-attack-graphs-to-analyze-social-engineering-threats/136466?camid=4v1a)