Chapter 6
Towards Designing FPGA-Based Systems by Refinement in B

Sergey Ostroumov
Åbo Akademi University, Finland

Elena Troubitsyna
Åbo Akademi University, Finland

Linas Laibinis
Åbo Akademi University, Finland

Vyacheslav Kharchenko
National Aerospace University KhAI, Ukraine

ABSTRACT

In this chapter, we propose a formal approach to designing FPGA-based systems. In particular, we introduce a general pattern for specifying synchronous systems and components as well as their typical interconnections. The proposed methodology for developing FPGA-based systems is based on the notion of refinement in the Event-B formalism. System development by refinement and proof-based verification provide the designers with powerful techniques for managing system complexity and achieving higher degree of system dependability. We aim at enabling a smooth transition from a formal Event-B specification to an implementable VHDL system representation. The proposed approach is illustrated by a case study – a development of an aircraft anti-icing system.

INTRODUCTION

Nowadays even quite complex embedded systems are often implemented on a single integrated circuit (chip). They are correspondingly referred to as systems on a chip (SoC). A particular kind of an integrated circuit, field-programmable gate array (FPGA), offers an attractive technology for implementing SoCs. However, complexity of FPGA-based systems makes their exhausting testing unfeasible and thus hinders their verification.

Yet, because of their use in critical applications, ensuring dependability of the FPGA-based systems remains a primary concern. Currently, the main means of achieving dependability are fault avoidance and fault tolerance. In our previous work...
we have analysed different forms of redundancy to achieve fault tolerance (Prokhorova, Kharchenko, Ostroumov, Yatsenko, Sidorenko, & Ostroumov, 2008). Here we focus on fault avoidance, specifically, on ensuring correctness of the FPGA-based systems via formal specification and proof-based verification.

In this chapter we demonstrate how to formalise stepwise development of the FPGA-based hardware systems by refinement in the Event-B formalism (Event-B and the Rodin Platform, 2008). We focus on development of synchronous hardware systems. Our approach enables verification of such systems at early stages of the system design as well as their correctness-preserving stepwise development.

In addition, in this chapter we propose a set of formal patterns for specification and refinement of generic hardware components as well as typical hardware assemblies in Event-B. In practice, hardware is usually described using a hardware description language, e.g., such as VHDL (Roth, 2007). In our approach we take this into account by showing how to translate the resulting Event-B specifications into the corresponding VHDL descriptions.

The chapter is organised as follows. Section “Background” introduces our formal specification language – Event-B – and also describes the basic notions used in modelling synchronous hardware. In Section “Patterns for Modelling FPGA-based Systems in Event-B” we put forward our approach by defining formal specification and refinement patterns for the FPGA-based systems. Moreover, we briefly explain how the resulting Event-B models could be translated into VHDL. Section “Case Study” exemplifies the proposed approach by a case study of an aircraft anti-icing system. Finally, in Section “Conclusion and Related Work” we discuss the results, overview the related work and propose future research directions.

**BACKGROUND**

**Modelling and Refinement in Event B**

Event-B is a state-based formalism for developing systems correct-by-construction (Abrial, 1996). The system behaviour is described using the Abstract Machine Notation (AMN). A system model (specification) consists of two parts, called a context and a machine (see Figure 1). The context defines the static part of the model – data types (sets), constants, and their properties given as a collection of axioms. The machine describes the dynamic behaviour of the system in terms of its state (model variables) and state transitions, called events. The essential system properties are formulated as invariant properties.

While refining an abstract specification, we introduce the required implementation details by adding new and/or replacing old data structures and events, thus bringing us closer to the eventual implementation. Proof of correctness of each refinement step is needed to establish that a more detailed machine refines its more abstract counterpart, while its new context extends the corresponding abstract context, as shown in Figure 2.
Related Content

True Color Image Segmentation by MUSIG Activation Function Using Self-Supervised QMLSONN Architecture With Context-Sensitive Thresholding
www.igi-global.com/chapter/true-color-image-segmentation-by-musig-activation-function-using-self-supervised-qmlsonn-architecture-with-context-sensitive-thresholding/202549?camid=4v1a

Cognitive Complexity Measures: An Analysis
www.igi-global.com/chapter/cognitive-complexity-measures/62538?camid=4v1a

A Test-Driven Approach to Behavioral Queries for Service Selection
www.igi-global.com/chapter/test-driven-approach-behavioral-queries/70732?camid=4v1a

Quality of Service in SDN Technology
www.igi-global.com/chapter/quality-of-service-in-sdn-technology/198199?camid=4v1a