Chapter 10

ON HARDWARE DESCRIPTION IN ECL

Lluís Ribas, Joaquin Saiz*

Microelectronics Unit – Computer Science Dept.
Autonomous University of Barcelona (UAB); Catalonia, Spain†
Lluis.Ribas@uab.es

Abstract ECL is a system design language suitable for heterogeneous, reactive systems, but there are no tools to directly compile the data-flow parts into hardware. In this paper, we propose several approaches to synthesizing these parts into hardware without extending the language. As a consequence, the resulting environment enables a true HW/SW co-design for small and medium-size systems.

Keywords: System specification languages, hardware description languages, Esterel, ECL, C-based SLDL

1. Introduction

System-level design languages (SLDL) must enable designers to express a broad range of behaviors so to describe a number of systems that can include several components with distinct characteristics (communication protocols, analog components such AD/DA converters, data-flow processes, et cetera.)

As nowadays’ hardware parts are able to perform complex tasks and they even include specialized functional units or embedded application-specific processors, they resemble processors that are used to execute software components. Therefore, preferred SLDL are inspired in programming languages, namely, C [8] or C++ [13]. (There are other SLDL based on other languages, too.)

*Supported by an UAB researchers’ grant.
†Partial funding provided by MCYT grant TIC2001-2508.
Of course, all SLDL do extend the original programming language they are based upon. SystemC [6], [7], [14], [15] takes profit of the object-oriented programming features to extend the base language with new operations and data types through the use of class libraries (one for each abstraction level and purpose).

On the other hand, other approaches extend the original language by new constructs supported by dedicated tools. For instance, HandelC [4] is a language based on a subset of ANSI-C with extensions to include parallel task definitions and communications via channels. However, it is not intended to fully cover the behavioral abstraction level. Rather, it is intended as a high-level abstraction description language for hardware. Also, SpecC [5] is defined as an extension to ANSI-C with constructs for implementing parallelism, synchronization, word-bit operation, and explicit FSM descriptions. (In HandelC, FSM are derived from the control-flow graph of the programs.) ECL [11, 10] belongs to this class of languages, as it is an extension to C language with constructs for inter-task communication and synchronization, and cycle-accurate semantics; which are derived from the Esterel [2] synchronous language. Also derived from it, there is no need to explicitly describe FSM.

Whatever the C-based SLDL, system descriptions are straightforwardly translated into programs that are, in fact, executable specifications. As a result, system simulations consist of specification executions. Likewise, software components are easily synthesized from specification. Consequently, their compilation into system processors’ machine languages is a direct step in the system design flow.

System specification usually starts at a very high level of abstraction where a few components are described. The design process can be viewed as a chain of specification refinements in which each step consists of adding detailed information to proceed further on the design flow until “implementable specifications” are generated. Ideally, a single SLDL should be used all the design flow through. This simplifies designers’ tasks and enables easy verification of each design stage (test benches from one design stage can be applied to the next, regardless of specific tests, which surely must be performed, too.)

As a result, any SLDL must be able to be used as a programming language or hardware description language (HDL), indistinctly. As for the latter, it must cope with RTL abstraction level constructs. At this level, EDA tools can be used to synthesize component descriptions targeted to hardware in a HW/SW system. The approach taken by SystemC requires using a dedicated class library and restricting specifications to the appropriate synthesizable subset of the language. Conversely, HandelC does not need such restrictions because of being, fundamentally,