ECL: A SPECIFICATION ENVIRONMENT FOR SYSTEM-LEVEL DESIGN

Gérard Berry
Ed Harcourt
Luciano Lavagno
Ellen Sentovich

Abstract We propose a new specification environment for system-level design called ECL. It combines the Esterel and C languages to provide a more versatile means for specifying heterogeneous designs. It can be viewed as the addition to C of explicit constructs from Esterel for concurrency and pre-emption, and thus makes these operations easier to specify and more apparent. An ECL specification is compiled into a reactive part (an extended finite state machine representing most of the ECL program), and a pure data looping part. The first can be robustly estimated and synthesized to hardware or software, while the second is implemented in software as specified. ECL is a good candidate for specification of new behavior in system-level design tools such as Cadence’s Cierto VCC tool[1]. ECL is especially targeted for specification of control protocols between data-computing behavioral blocks.

1. OBJECTIVES

System-level designs are typically conceived as a set of communicating processes. The processes may communicate synchronously or asynchronously, may be control- or data-dominated, may have hard real-time constraints, and may be used in embedded systems. Such a wide variety of characteristics and requirements implies that there is no single language that can be efficient for specification. Nonetheless, it is desirable to be able to specify such designs in an integrated environment, so that the design as a whole can be both treated with a common semantics, at least at the communication level, and automatically synthesized, at least to the extent possible.

For this reason, we propose the use of a new executable specification environment called ECL. The main idea is to combine two existing languages to create a specification medium that can benefit from the features of both languages and their existing well-developed compilers. In particular, we add the convenient and concise constructs from Esterel for concurrency and pre-emption to C.

A prototype ECL compiler has been completed and is currently being tested and further developed on some industrial examples.

2. BACKGROUND

2.1 ESTEREL

Esterel [5, 4] is a language and compiler with synchronous semantics. This means that an Esterel program has a global clock, and each module in the program reacts at each “tick” of the global clock. All modules react simultaneously and instantaneously, computing and emitting their outputs in “zero time”, and then are quiescent until the next clock tick. This is classical finite state machine (FSM) behavior, but with a description that is distributed and implicit, making it very efficient. This underlying FSM behavior implies that the well-developed set of algorithms pertaining to FSMs can be applied to Esterel programs. Thus, one can perform property verification, implementation verification, and a battery of logic optimization algorithms.

The Esterel language provides special constructs that make the specification of complex control structures very natural. It is often referred to as a reactive language, since it is intended for control-dominated systems where continuous reaction to the environment is required. Communication is done by broadcasting signals, and a number of constructs are provided for manipulating these signals and supporting concurrency and signal pre-emption (e.g., parallel, abortion and suspension).

The Esterel compiler resolves the internal communication between modules, and creates a C program implementing the underlying FSM behavior. A sophisticated graphical source-level debugger is provided with the Esterel environment. While Esterel only provides a few simple data types, one can create and use any legal C data types; however, this is separate from the Esterel program, and must be defined separately by the designer. Pure C procedures and functions can be defined by the user and called from an Esterel program, but again there are definitions and code that must be written by hand by the designer.