Proposed Extensions to VHDL for Abstraction of Concurrency and Communication

Peter J. Ashenden <petera@cs.adelaide.edu.au>

Philip A. Wilsey <phil.wilsey@uc.edu>

Technical Report TR-97-11

Department of Computer Science
The University of Adelaide, SA 5005
Australia

Technical Report TR-210/12/97/ECECS

Department of Electrical and Computer Engineering
and Computer Science
University of Cincinnati,
PO Box 210030
Cincinnati, OH 45221–0030
USA

This work was partially supported by Wright Laboratory
under USAF contract F33615–95–C–1638

Abstract

This report describes extensions to VHDL to support system-level behavioral modeling by providing more abstract forms of communication and concurrency than those currently in the language. The report summarizes design objectives and issues that must be considered in developing such extensions, and presents detailed definitions of our extensions. The extensions for communication consist of channel types, channel objects, dynamically allocated channels, and message passing statements. The extensions for concurrency consist of process declarations and static and dynamic process instantiation statements. Use of the extensions is illustrated with examples.

1. Introduction

As the complexity of integrated hardware and software systems increases, system-level design languages are becoming increasingly important. Such languages rely on abstraction as the key to managing complexity. Designers focus first on the abstract properties of a system in various domains and devise a systems architecture that will satisfy the requirements placed on the system. The domains under consideration include behavior, structure, performance, physical arrangement and packaging, power consumption, thermal, cost, and so on. In each domain, abstraction is used to focus on the major aspects of the system and minor detail is ignored. Judicious choice of abstractions makes architectural design and analysis tractable, and aids subsequent partitioning and refinement of the system design.

Hardware description languages focus on describing systems in the behavioral and structural domains. However, due to their origin as languages for hardware design, they frequently do not include strong capabilities for abstracting over data and for describing complex interactions. For example, in Verilog [13, 17], data types are closely bound to their binary representation, and signalling between modules includes aspects of electrical implementation. VHDL [1, 12], on the other hand, allows more abstract expression of data, and its type system is similar to that of conventional programming languages. However, its signalling features are still closely bound to electrical implem