Accepted for publication in *Proceedings of Sixth International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS '98).*

Extensions to VHDL for Abstraction of Concurrency and Communication*

Peter J. Ashenden
Dept. Computer Science
University of Adelaide, SA 5005
Australia
petera@cs.adelaide.edu.au

Philip A. Wilsey
Dept. ECECS, PO Box 210030
University of Cincinnati
Cincinnati, OH 45221-0030, USA
phil.wilsey@uc.edu

Abstract

This paper 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 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 [14, 18], data types are closely bound to their binary representation, and signalling between modules includes aspects of electrical implementation. VHDL [1, 13], 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 implementation.

To remedy these deficiencies, we have developed extensions to VHDL to improve its support for system-level modeling. These extensions are based on the requirement in a system-level description language for abstraction in the following areas:

- abstraction of data,
- abstraction of concurrency, and
- abstraction of communication and timing.

These extensions make VHDL suitable for describing at an abstract level aspects of a system that may ultimately be implemented in hardware or software. We have described the details of the extensions to the data modeling facilities in previous papers [4–6]. These extensions involve mechanisms for object-oriented data types and for genericity. We have also presented a discussion of the issues that must be considered in extending VHDL to provide more abstract forms of concurrency and communication [3].

---

* This work was partially supported by Wright Laboratory under USAF contract F33615-95-C-1638.

In this paper we present our extensions for abstraction of communication and concurrency in the SAUVE (SAVANT and University of Adelaide VHDL Extensions) project. We introduce into VHDL the notions of communication channels and message-passing operations as an abstraction of communication by signals. We also extend the process model by allowing process declarations that can be statically or dynamically instantiated.

Section 2 of this paper reviews our design objectives. Section 3 discusses issues that must be considered in extending VHDL with a more abstract form of communication and gives reasons for our choices among the alternaitves. Section 4 presents the details of the abstract communication language features in SUAVE and Section 5 presents the details of the extensions for concurrency abstraction. Section 6 presents an extended example, a multithreaded client-server system, that illustrates the combined use of the extensions in a system-level model. Finally, Section 7 contains our conclusions.

2. Design Objectives

Our main design objective in the SUAVE project is to improve high-level modeling support in VHDL through increased use of abstraction. Specific objectives leading to the extensions described in this paper are:

- to provide a more abstract form of comm