Assertion-Based Design with Horus

Yann Oddos, Katell Morin-Allory and Dominique Borrione
Laboratoire TIMA, INPG/UJF/CNRS, 46 avenue F. Viallet, 38031 Grenoble cedex, France.
E-mail: name.surname@imag.fr

I. INTRODUCTION

The Horus project aims at providing methodologies and tools for efficiently supporting property-based design all along the design flow [FKL03]. Formal properties can be expressed at the earliest specification step, experimented with, refined and complemented, using standard languages with formal semantics [IEE05a], [IEE05b]. In this paper, PSL is used, but the approach applies to SVA as well. Properties may express:

- The results expected from the device. These assert statements can be checked throughout the design flow using observation monitors.
- The characteristics of the device environment, under the form of constraints on its inputs. Input sequences satisfying these assume statements can be produced by test sequence generators.

From asserted and assumed properties, Horus automatically produces synthesizable source (VHDL or Verilog) modules for the corresponding monitors and generators. It also provides a graphical user interface to support the interconnection of these modules with the design at hand. Note that the same modules, without any adjustment, can be used for simulation, emulation, or synthesis purposes, with the objective of design verification, system integration test, and online fault detection during execution.

With respect to previous works and existing CAD software, Horus exhibits new and competitive advantages:

- The monitors and generators are built in a modular way, so that modules for sub-properties can be reused in more complex ones.
- The modular construction is very efficient: it takes a fraction of a second for dozens of complex properties.
- The method for building the monitors and the generators has been proven correct with the PVS theorem prover [MAB05].

II. PRINCIPLES OF GENERATOR AND MONITOR SYNTHESIS

Our work is based on the formal trace semantics of the operators given in [IEE05a]. In this paper, we assume the design and the properties to be clock synchronized.

The monitor and generator interfaces have a common set of input ports: the clock, a reset and a start signal. A monitor takes as additional inputs the observed signals. Two output signals valid and pending (for strong operators) display the different PSL satisfaction levels (holds, holds strongly, pending, fail). Figure 1 shows the interface signals for the monitor of property P.

A generator takes as additional outputs the signals involved in the property and a signal pending used to indicate the status of the generated signals.

The monitor and generator synthesis relies on:

- A library of primitive elements, one for each PSL operator. The libraries are distinct for the generators and the monitors (see Figure 2).
- A systematic connection procedure to build complex monitors or complex generators from primitive ones, based on the PSL expression syntax tree.

Fig. 2. Generic architecture of a primitive component

III. THE HORUS PLATFORM

The Horus software is based on two parts: the core implemented in C and the graphic interface implemented in Java. The core implements a PSL parser, generates the VHDL or Verilog description of the property. It is approximately 10000 lines long.

The HORUS environment helps the user build an instrumented design to ease debugging: it can synthesize monitors and generators, connect them to the design under test (DUT) and adds a device to snoop the signals of interest. It comes with the VHDL and Verilog flavors. The HORUS system has a friendly GUI for the generation of the instrumented design in 4 steps (see Figure 3).

Step 1 - Design selection: The DUT, with its hierarchy, is retrieved.

Step 2 - Generators and Monitors synthesis: Select properties or property files, define new properties, select target HDL language, synthesize monitor or generator.

Step 3 - Signal interconnection: The user connects the monitors and the generators to the design. All the signals and variables involved in the DUT are accessible in a hierarchical way. The user needs only select the signals to be connected to each verification IP.

Step 4 - Generation: The design instrumented with the verification IPs is generated. When internal signals are monitored,
the initial design is slightly modified to make these signals accessible to the monitors.

The instrumented design has a generic interface defined for an Avalon or a Wishbone bus. If the FPGA is based on such a bus, the user can directly synthesize and prototype the instrumented design on a FPGA. The platform has been applied to several case studies among which a Wishbone compliant crossbar [OPE]. The conmax_ip controller [Uss] allows the communications between up to 8 masters and up to 16 slaves on a crossbar switch with up to 4 levels of priority. Many aspects of the conmax_ip have been verified: the initialization upon reset, the priority and round robin policies, the correct connection between masters and slaves, etc. To test the correctness of the conmax_ip controller in isolation, without the overhead of simulating a complete set of masters and slaves, the environment that provides correct test signals was modeled with generators compliant with the hand-shake protocol.

The test platform is illustrated on Figure 4. It was synthesized with QuartusII 6.0 on an Altera DE2 board with a CycloneII EP2C35 FPGA chip set.

IV. CONCLUSIONS AND PERSPECTIVES

The Horus tool, based on formally proven correct methods, provides a unified support to assertion-based design, between the specification and the test phases. Given a set of logical and temporal properties written in PSL, Horus automatically constructs a test environment for the design. This construction is fast, correct, and produces efficient monitors and generators. The size of the instrumented design is determined by the number of distinct properties needed to specify the behavior and by the number of repetitions of each property over duplicated blocks that play symmetric roles. We have seen in the case of a wishbone switch that the number of repetitions may be quadratic in the number of nodes that compete for a resource, times the number of resources.

The main advantages of our tool is to cover the whole PSL simple subset, and the whole verification flow: from the simulation to the on-line testing. It can relieve the test engineer from the tedious work of writing correct test benches. When synthesized on FPGA, the instrumented design under test can execute at full speed. The status of each property in the instrumented DUT (strongly hold, hold, pending, fail), displayed on the pending and valid outputs of each monitor, can be traced at each clock cycle. In case of a failed property, the signals involved and the precise temporal origin of their sub-trace are identified. User-friendly post-processing debug aids, such as graphical trace displays, are not currently available in our University prototype, but are planned in future developments in cooperation with industrial partners.

REFERENCES

[OPE] OPENCORE.ORG and Silicore. WISHBONE System-on-Chip (SoC) Interconnection Architecture for portable IP Cores.
[Uss] R. Usselman. WISHBONE Interconnect Matric IP Core.