A tool-set for simulating Altera-PLDs using VHDL

André Klindworth

University of Hamburg, Dept. of Computer Science
Vogt-Koelln-Str. 30, D-22527 Hamburg, Germany

Abstract. This paper presents a tool-set for simulating Altera-PLDs [1] using VHDL [2]. It has been successfully used in a graduate course on digital design with PLDs. The tool-set supports timing simulation as well as functional simulation of designs that have been designed with the Altera MAX+plusII development tool.

1 Background

Altera's PLD/FPGA design tool MAX+plusII comes with an own hardware description language (HDL) named AHDL [3]. Compared to standard HDLs like VHDL, AHDL is a low-level HDL with support for signals of type either bit or vector-of-bits only. The implicit signal types along with short-hand constructs for instantiating and interfacing of subdesigns allows very compact hierarchical design descriptions. In addition, AHDL gives the user a direct control over technology mapping and design partitioning. Both aspects let us choose AHDL as the means of design entry in a first year graduate course on digital design for PLDs in which most participants have little or no experience in using HDLs as well as PLDs.

2 Simulation Requirements

Validation by simulation is still the usual way verification of digital designs is done. The designer should have easy access to a powerful simulator and the development tools of PLD-vendors usually come with programs which generate a design description in a standard format like EDIF, Verilog, and/or VHDL. Such a description then can serve as an input to a third-party simulator.

When simulation is considered, it is highly desirable that the designers view of the design is structured in the same way as in the original design description, that is, design hierarchy, groups of signals (busses), and all symbolic names of instances, machine states, and internal as well as external signals should be preserved. MAX+plusII for workstations fails to meet these demands.[1] Netlist writers that produce an EDIF, VHDL or Verilog description of a design are available, but they merely write flattened netlists. These netlists contain full

[1] The PC-version of MAX+plusII comes with an integrated simulator, but the expressive power of the stimulus description language is limited.

timing information, but neither the design hierarchy nor names (except for the primary inputs and outputs) are preserved. Functional debugging of a design has to be done by analysis of the primary inputs and outputs only.

But before correct timing of a circuit is to be verified, the designer must check the functionality of the design. For this task, the exact timing of the circuit is quite irrelevant and a *functional* simulation suffices. This is especially true when the design is fully synchronous, as it is enforced by the structure of functional blocks in most PLDs. We believe that at this stage of the design process it is much more important to simulate a design that "looks" exactly the same as in the original description rather than using a modified model of the design with exact timing information.

3 Simulation Procedure

As a compromise between the needs of the designer and what can be fulfilled with a reasonable effort, we developped a tool-set which eases timing simulation and provides functional simulation of Altera PLDs that have been designed using AHDL and the MAX+plusII compiler. For both types of simulation, VHDL is used to interface MAX+plusII with a third-party simulator which in our case is the powerful Synopsys VHDL-simulator *vhdlsim* [4]. VHDL is also used by the user to define stimuli for the simulation. For this purpose, our tool-set embeds the design-to-test in an automatically generated VHDL testbench. This testbench is a VHDL entity with an architecture that contains a component declaration for the design, a signal declaration for each of its external connections, and an instantiation of the design, the ports of which are connected to the signals. Using such a testbench has been proposed in [5] and gives the user the full expressive power of VHDL to assign waveforms to the external inputs and to observe and evaluate the resulting responses of the design.

Our tool-set supports both types of simulation: *timing simulation* and *functional simulation*.

For *timing simulation*, the VHDL netlist written by the MAX+plusII compiler is slightly modified and embedded in the testbench. Two script files named *timegen* and *tsim* make the necessary calls to programs which

- check that the design project is up-to-date and has been successfully compiled in MAX+plusII.
- modify the VHDL-netlist that has been written by MAX+plusII. Modifications include regrouping of busses that have been resolved by the compiler.
- generate a VHDL testbench for the design.
- make the calls to the VHDL compiler to generate simulation models for the VHDL description of the design itself and its testbench.
- call the simulator.

For *functional simulation*, a VHDL model of the design is generated directly from the original AHDL description. This task is carried out by an AHDL-to-VHDL compiler named *a2vhdl*. The VHDL code produced by *a2vhdl* fully preserves the hierarchy of the design project as well as all signal groups (busses) and all symbolic names of instances, machine states and signals. This VHDL description of the design is embedded in the same testbench as the timing model, so that the same stimuli are used for both types of simulation. Changing between functional and timing simulation is done by simply configuring the testbench for using the timing architecture or the functional architecture of the design, respectively. Again, two script files named *funcgen* and *fsim* realize all the necessary steps to simulate an AHDL design that has been succesfully compiled with MAX+plusII.

To simulate an AHDL design using our tool-set, the designer has to follow the following procedure:

1. Design entry with AHDL.
2. Compile design with MAX+plusII.
3. Call *funcgen* or *timegen*, respectively.
4. Insert stimuli processes in the testbench.
5. Call *fsim* or *tsim*, respectively, to start the simulation.

The VHDL code generated for the design is transparent to the user. When doing a timing simulation, all he sees from the VHDL model is the testbench, containing the external connections of the design. In the case functional simulation is chosen, the same testbench is used. In addition, the interior of the functional VHDL model is structured in the same way as the original AHDL description. All design objects the user defined in the original design description do also exist in the functional VHDL model and have the same name. Internal signals may be traced during simulation using the original (hierarchical) node identifiers.

4 Conclusions

The tool-set as described in this paper proved to greatly simplify debugging of PLD designs. It combines the compactness of design description in AHDL with the expressive power of VHDL for the description of simulation stimuli and the monitoring capabilities of a powerful VHDL simulator.

References

1. *Altera Databook*, Altera Corporation, San Jose CA, 1993.
2. *IEEE Standard VHDL Language Reference Manual*, IEEE Standard 1076-1987, New York, 1988.
3. *Altera MAX+plusII reference manual*, Altera Corporation, San Jose CA, 1992.
4. *Synopsys VHDL Simulator Reference Manual*, Synopsys Inc., 1993.
5. Z. Navabi: *VHDL - Analysis and Modelling of Digital Systems*, McGraw-Hill, New York, 1993.