The Increasing Importance of HDL Verification

OK
Not OK

How the new generation of HDL simulators can help you design the largest FPGAs with a minimum amount of time spent on the simulation process.

by Gregor Siwinski, Director of R&D, Aldec Inc., gregor@aldec.com

When Xilinx users first started creating FPGA designs based on schematics, the only verification technology available was based on a gate-level netlist simulator. The time and effort required to create tool-specific simulation test vectors and the relatively small gate count of designs made it unnecessary to use a simulator in the design flow. It was easier to program the device and test it in hardware than to verify it using the simulator.

Today, with FPGA capacities exceeding one million system gates, simulators can actually save you time by detecting problems early in the design flow. For most large designs it is practically impossible to create a reliable FPGA without using a simulator.

Behavioral RTL Simulation of HDL Code

The term "behavioral RTL simulation" is used here to describe simulation of an FPGA design prior to running any synthesis or implementation tools. Behavioral simulation verifies that your HDL code is correct and detects any functional problems. For large designs that take hours to synthesize it will save a lot of time if the FPGA can be functionally tested without running the synthesis after each change. Furthermore, behavioral simulation is typically 10 to 100 times faster than post-synthesis simulation of the same design.

Today, with FPGA capacities exceeding one million system gates, simulators can actually save you time by detecting problems early in the design flow.

Advanced HDL Debugging

Simulating HDL in a debug mode allows you to analyze your design source similar to software debuggers:

- Execute your HDL code in trace mode one line at a time.
- Breakpoints can be on any line of HDL code or signal change.
- Variables can be monitored and modified during simulation.
- Design hierarchy and signals can be viewed to inspect a value of any signal or port in the design.
- The Dataflow view presents a graphical view of a signals connectivity in the HDL design.
- Waveforms and event list windows can be used to monitor the results.

Continued on the following page

HDL simulation in debug mode using Active-VHDL™ software

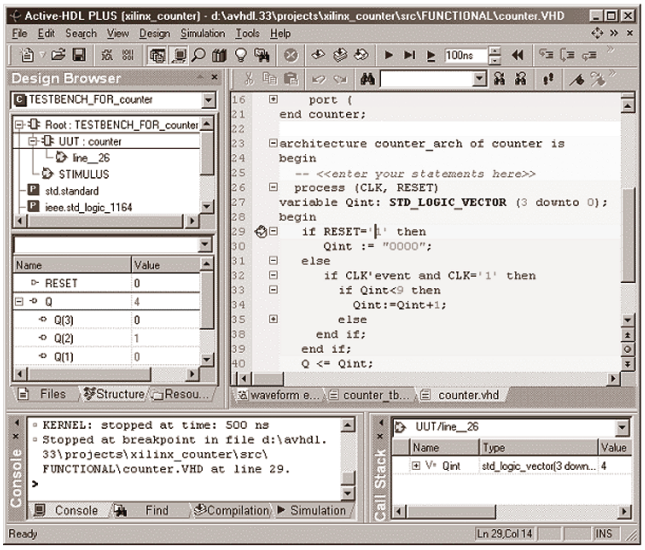

Figure 1

In most cases, the simulation of a design after synthesis should output the same results as the behavioral simulation. The purpose of post-synthesis simulation is to make sure this is the case...

HDL Test Bench

Creating simulation input can be a tedious process. That's why it is important to know all possible entry methods and use them to save design time:

- Graphical stimulators are very useful to set signals to a desired state, define clocks and formulas.
- HDL test bench files can be generated from waveform diagrams.
- HDL test bench with Smart Comparison™ using IEEE 1029.1 WAVES standard.
- Script files can automate the entire verification process (such as TCL, PERL).

Waveform-based HDL simulation with familiar stimulators

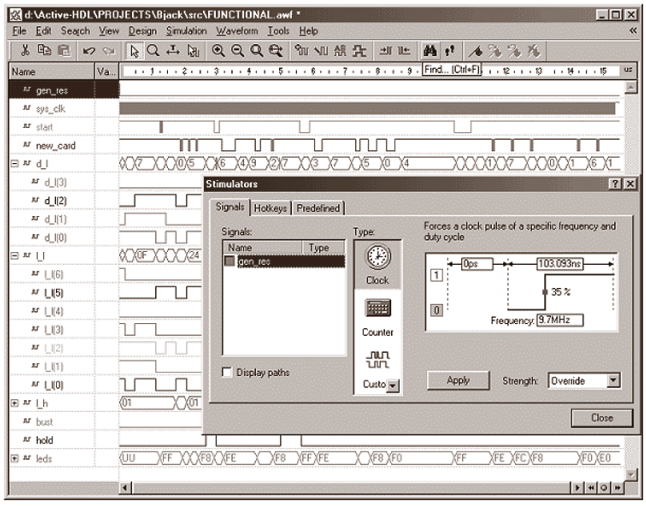

Figure 2

One very important benefit of using HDL simulation is that the test bench can be described in the same HDL language as the design itself. Aldec also generates a script file that will compile the necessary HDL files and automate the entire verification process.

Xilinx Foundation Series Designs

Many Xilinx Foundation Series software users create designs containing schematics. Schematic-based designs and mixed schematic/HDL designs can also be simulated in HDL simulators. The Foundation Series software provides a seamless interface to HDL simulation that will export all schematic portions of the design to VHDL and then invoke the VHDL simulator from the Foundation Project Manager.

Users who create graphical state machines in the Foundation Series are also able to animate the FSM diagrams during simulation. Also, the Foundation test vectors can be imported to Active-VHDL for easy transition to the HDL simulation environment.

Post-synthesis Verification

In most cases, the simulation of a design after synthesis should output the same results as the behavioral simulation. The purpose of post-synthesis simulation is to make sure this is the case and that the synthesis tool output produced the netlist which is functionally the same. Synthesis programs may implement your HDL code in a different way than you expected and the post-synthesis simulation will detect that.

Synthesis tools like Foundation Express can produce a netlist in VHDL and Verilog formats for HDL verification. The best feature of this process is the ability to use the same simulation input or test bench that was used for behavioral simulation. The post-synthesis simulation results can be compared against original outputs graphically in the waveform window.

Xilinx Foundation software
provides integration to VHDL simulation

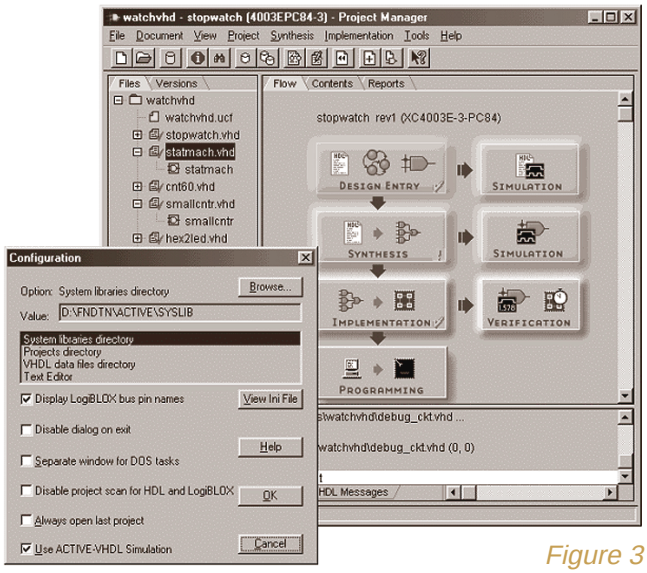

Figure 3

Timing Verification

After a design is implemented by the Xilinx tools, you have an option to export timing simulation data. This will generate an HDL simulation netlist and an SDF file containing calculated timing delays. The timing netlist uses the SIMPRIMS simulation library based on Vital primitives.

Accurate simulation of timing delays allows testing of the design functionality at the target frequency rate, to detect any setup/hold violation and timing glitches.

Similar to post-synthesis simulation, there is no need to develop new simulation test vectors. The same simulation test bench can be used for both behavioral and post-implementation simulation. All this is done in the same design and graphical environment.

Timing simulation can be very time consuming. However, with the accelerated Vital primitives used in Active-VHDL software the simulation speed can be five to ten times faster than in the Foundation Series gate-level simulator.

In the near future, you will see even more integration in the HDL design environment with features such as HDL graphical entry, code coverage analysis, and formal verification. This prepares us for the next generation of HDL verification tools.

Smart Comparison Test Bench

One obvious question is how to compare the simulation results between behavioral simulation and post-implementation timing simulation. You can no longer use the graphical comparison because of the delays that cause waveforms to shift.

The answer comes in the Smart Comparison based on the recently published IEEE WAVES standard. Among other test bench functions, WAVES provides a very convenient method of comparison of current simulation results with the golden reference previously saved in the vector file.

Active-VHDL Test Bench Wizard™ automates this process even further, saving the functional simulation results into a standard VEC file and generating the VHDL test bench program using the WAVES library. You only have to specify the so called "comparison window" which defines a period of time after each test pattern that is ignored to take account for timing delays. All discrepancies are detected during simulation giving you detailed messages about signal and time when the simulation results are different. Both expected and actual waveforms are displayed in the same screen.

Smart Comparison shows
actual and expected waveforms

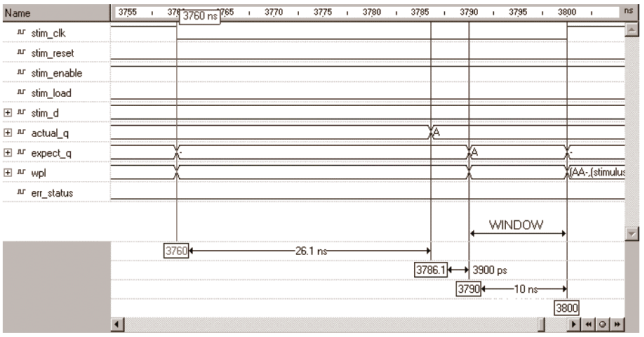

Figure 4

Summary

As you could see, the verification tools have come a long way over the last 15 years. If you would like to learn more about the benefits of HDL simulation please visit Aldec's website at http://www.aldec.com/activevhdl

In the near future, you will see even more integration in the HDL design environment with features such as HDL graphical entry, code coverage analysis, and formal verification. This prepares us for the next generation of HDL verification tools.