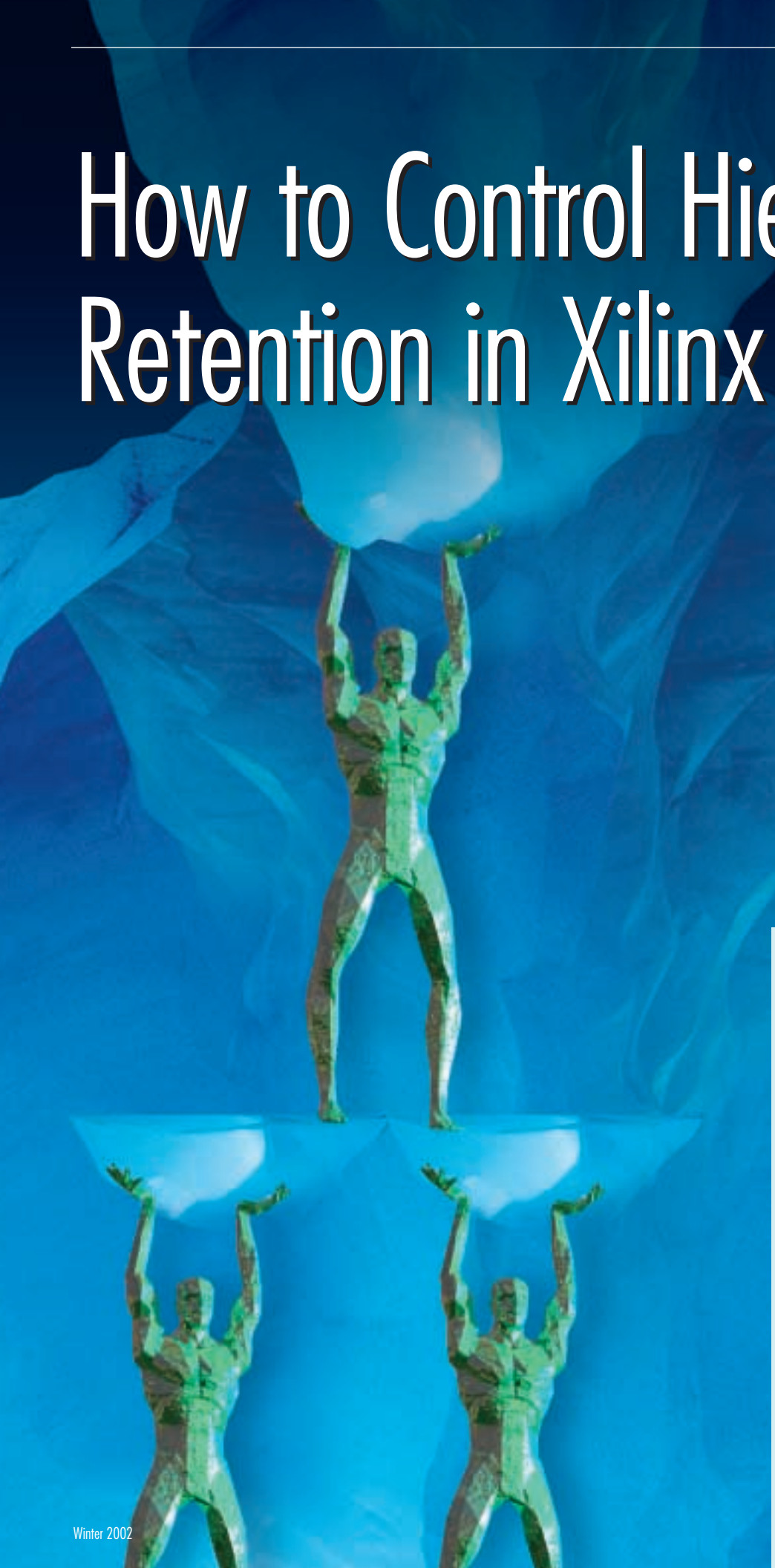
How to Control Hierarchy Retention in Xilinx ISE 5.1i



With ISE 5.1i, you can break down your design into a hierarchical structure of modules, reducing the complexity of construction, analysis, and verification.

by Brian Philofsky
Software Technical Marketing Engineer
Xilinx, Inc.
brian.philofsky@xilinx.com

The hierarchical modularity typical of modern PLD designs contributes substantially to the efficiency and reliability of front-end RTL design verification. However, for earlier versions of the Xilinx ISE, preserving this hierarchy information through the design flow was problematic. Much of this hierarchy information was often lost by the time the design flow reached the timing analysis and back-end verification. Without this information, these verification steps had to rely on a monolithic rather than a modular examination of the structural design. This verification method was inefficient, more error-prone, and time consuming.

Xilinx ISE Release 5.1i implements several improvements throughout the design flow to permit you to select between exact retention of design hierarchy for those areas of the design where ease of analysis is paramount. ISE 5.1i also allows hierarchy flattening when the design demands optimization across hierarchy boundaries. For each area of the design, you can employ the optimum tradeoff between design visibility for back-end verification, and the best possible performance and area optimization.

5.1i Design Flow Enhancements

Modern PLD designs are typically so complex that it is unrealistic to design and verify them as monolithic objects. Breaking the design down into a hierarchical structure of modules reduces the complexity of construction, analysis, and verification to manageable levels. Even less complex designs routinely benefit from hierarchical structure with improved understanding, documentation, and code reusability.

In previous versions of the Xilinx ISE software, hierarchy retention was an optional switch for simulation netlist creation. However, the actual hierarchy created did not always correlate well to the input design hierarchy because of synthesis and place-and-route (PAR) tool optimizations. Although some boundary optimizations can be controlled from the synthesis tool with the use of synthesis directives or global optimizations, this information was never communicated to PAR, which, as a result, was unable to determine which modules were intended and preserved by synthesis and thus should be preserved and recreated for post-PAR simulation.

Also, a synthesized design often contained both user-created and unintended hierarchies. Unintended hierarchies were created by mechanisms such as generate statements, primitive instantiations, and third-party IP hierarchy. Xilinx ISE 5.1i expands the use of an existing user attribute, KEEP_HIERARCHY, to communicate to the back-end PAR tools which hierarchies were preserved in synthesis and are intended to be preserved throughout the tool flow. This ensures the hierarchy that improves design verification will be preserved – and the hierarchy that does not improve design verification – will be flattened for improved path optimization. Figure 1 illustrates how the design flow has changed in the 5.1i release.

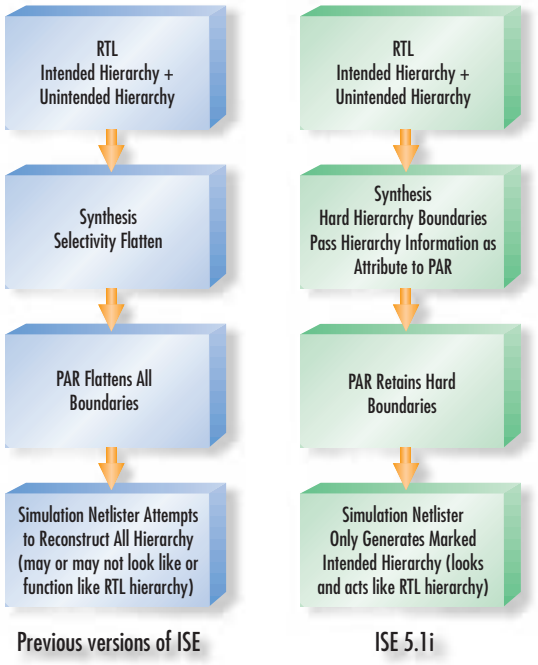


Figure 1 - Flow changes in ISE 5.1 for hierarchy retention compared to previous versions of the software

How to Use KEEP_HIERARCHY Attribute

The current version of the Xilinx Synthesis Tool (XST), as well as future releases of Synplicity and other synthesis tools, will pass the KEEP_HIERARCHY attribute automatically when hierarchy is preserved during synthesis.

For versions that do not currently pass the attribute automatically, the KEEP_HIERARCHY attribute may be passed manually, within the RTL code, within a synthesis constraint file, or within a user constraints file (UCF). Table 1 shows example syntax for passing the attribute.

For alternative flows, such as incremental design, modular design, or any bottom-up synthesis flow that generates separate EDIF files for each level of hierarchy, use the -insert_keep_hierarchy switch in Ngdbuild to automatically place the KEEP_HIERARCHY attribute on each input EDIF file.

For the ISE Project Navigator interface, you can specify the KEEP_HIERARCHY attribute by selecting the "Preserve Hierarchy on Sub Module" option in the "Advanced Process" menu. Figure 2 shows the location of this switch in the ISE Project Navigator tools.

The KEEP_HIERARCHY methodology works exceptionally well with these design flows by supporting modular design techniques that

Example UCF syntax (assuming hierarchy was preserved during synthesis):
INST *hierarchy_name* KEEP_HIERARCHY=TRUE;

Example of Synplicity SDC file syntax:
define_attribute {v:*module_name*} syn_hier {hard}
define_attribute { v:*module_name*} xc_props {KEEP_HIERARCHY =TRUE}

Example of Synplicity Verilog code syntax (module instantiation):
module_name instance_name(*port_mapping*) /* synthesis syn_hier="hard"
xc_props="KEEP_HIERARCHY=TRUE" */;

Example Synplicity VHDL code syntax (placed in architecture of preserved hierarchy):
attribute syn_hier : string;
attribute syn_hier of *architecture_name*: architecture is "hard";
attribute xc_props : string;
attribute xc_props of *architecture_name*: architecture is "KEEP_HIERARCHY=TRUE";

Table 1 - Example syntax of how to pass KEEP_HIERARCHY *for Synplicity*

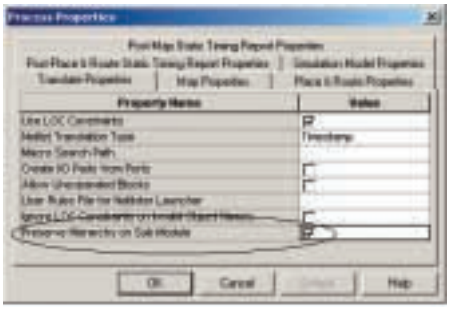


Figure 2 - Location of "Preserve Hierarchy on Submodule" option in ISE 5.1i

lend themselves to incremental design creation, implementation, and verification.

Once the attribute is placed on a module, the back-end PAR tools will preserve the instance names, port declarations, and logic functions within the hierarchy boundary. The resulting simulation netlist produced by the VHDL or Verilog™ netlister will contain the desired hierarchy structure for back-end timing and functional verification.

Convergence of Timing and RTL Simulation Methodologies

An important benefit of the new KEEP_HIERARCHY feature is the similarity that it establishes between the front-end and back-end verification methodologies. The structural simulation timing netlist may be loaded into any HDL simulator and used similarly to the RTL netlist. Hierarchical forces, breakpoints, probes, and watch points created during RTL simulation should exist at all hierarchical ports.

Simulator scripts, the waveform viewer, or the testbench itself may reference hierarchical signals just as in the RTL simulation. Testbenches and RTL simulation scripts should be generally reusable. Identifying signal locations in the hierarchy browser of the simulator should be easier and schematic views of the structural design should look similar to those in the original design source code.

This new methodology should also improve the productivity of structural timing simulations. Figures 3 and 4 show how the hierarchical netlists can be used in popular simulators to improve back-end verification.

Conclusion

As PLD design size and complexity increase, design verification is becoming a significant bottleneck within the design process. Xilinx ISE 5.1i will dramatically improve your control over hierarchy retention, which will help alleviate this bottleneck by increasing the efficiency of verification and shortening the verification cycle for PLD designs.

The new ISE 5.1i software products are now available for purchase at the ISE website at *www.xilinx.com/ise/*.

Further improvements to this feature are planned for future releases of Xilinx ISE software, including the ability to write out separate structural netlists and SDF files for each level of preserved hierarchy, allowing for true modular verification of the design.

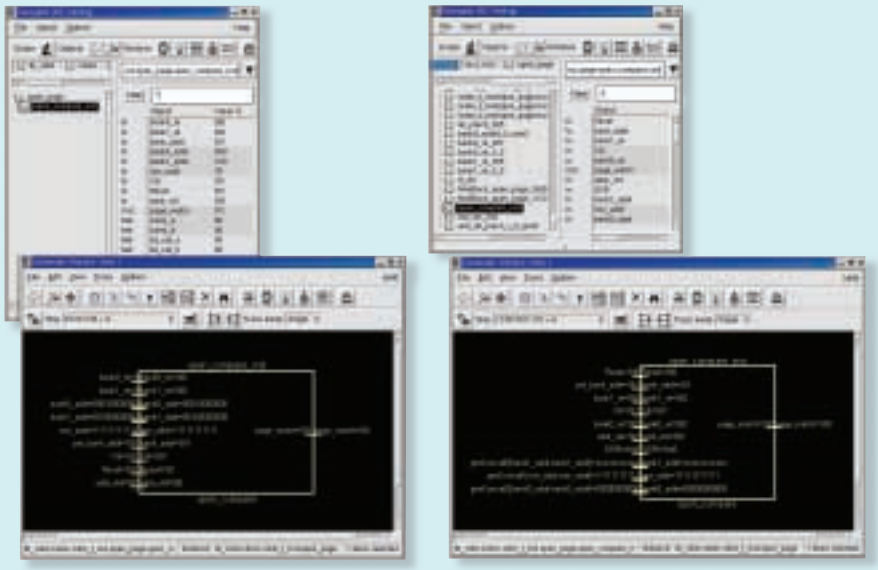


Figure 3 – Views of a sub-level module in Cadence NC-Sim Navigator window and Schematic window; left side is RTL design and right side depicts structural view of same design.

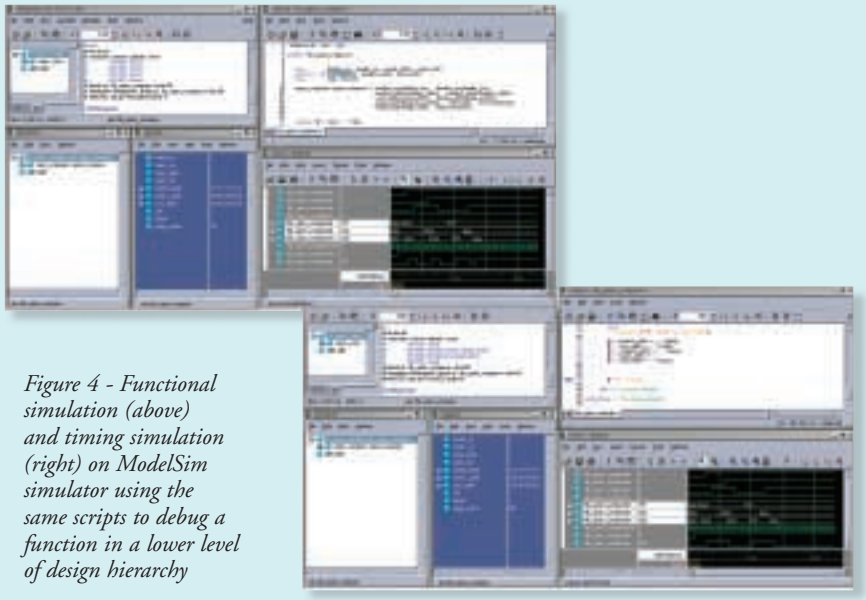


Figure 4 - Functional simulation (above) and timing simulation (right) on ModelSim simulator using the same scripts to debug a function in a lower level of design hierarchy