UNIVERS HDL NAVIGATOR

All design details easily revealed

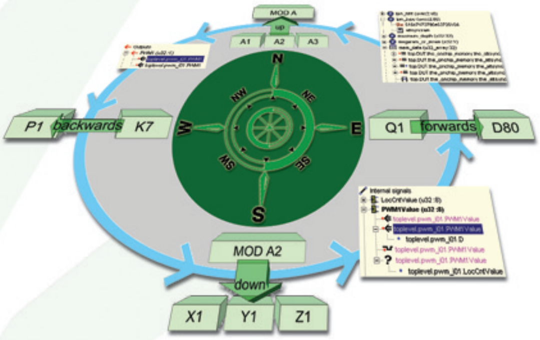
INTRODUCTION

The Univers HDL Navigator is the perfect tool to inspect and analyze any HDL code. Even if the HDL code is known by the engineer, the HDL Navigator is a big time saver. This tool uses both the source code and the compiled result, which has many advantages. Often a signal is renamed when it enters a module instantiation, a text search tool would not be able to follow that signal further; the search string must be manually changed. When the compiled result is used the tool looks through such time consuming and error prone difficulties.

All design details are presented in a practical GUI for user convenience. This GUI shows: the design hierarchy and the following items of a selected module: Source code, IO signals, internal signals and parameters. Many browse features exist to follow signals forwards or backwards through the design hierarchy. The HDL Navigator has also a direct link to other Univers windows, for instance to add a selected signal in the Signals window to view the waveform.

Features:

- Easy to use
- Shows actual design hierarchy
- Maintains hierarchy when signal info is presented
- Looks through signal renaming and extractions
- Shows resolved parameter values
- Shows signal dependencies and usage
- Browses forwards and backwards
- Recognizes constants, memories, assignments, etc
- Direct link to Signals window (waveform view)
- Direct link to the Simulator (actual signal values)
- Supports Verilog IEEE Std 1364-1995 and 1364-2001



DESIGN INSIGHT

Using this HDL Navigator easily answers common questions during design analysis and bug tracing:

- "What is the value of this parameter in this instance?"
- "Which signals contribute to the generation of this signal and where do they come from?"
- "What are all the destinations of this signal?"
- "What hierarchy is hidden in this design?"

The HDL Navigator provides full insight in a design. Using the HDL Navigator has proven to incredibly boost the process of finding bugs in any HDL design. It is extremely powerful in tracing an misbehaving signal backwards in the design and to show all the dependencies of that signal. Suspected contributing signals are directly assigned to the Signals window to view the waveforms. Dependent on the waveforms the misbehaving contributing signal is further traced backwards to the source of the bug.

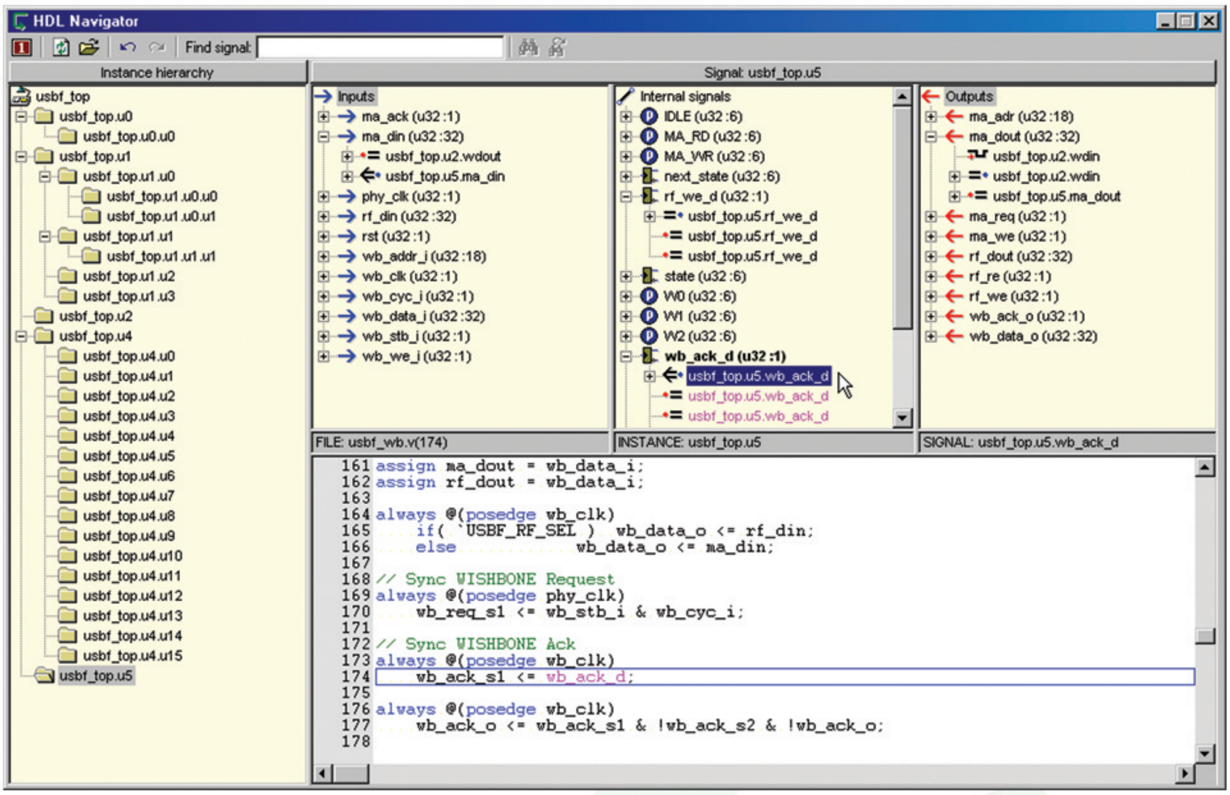
NAVIGATOR VIEWS

The main window is split up into several areas, providing a total view of a selectable design module. In the leftmost pane the design hierarchy is shown. In this hierarchy a module can be selected. The rest of the window shows in three panes: the 'input signals', 'internal signals' and 'output signals'. Beneath this information the source code of the selected module is shown.

Browsing is supported by three methods: Select a module in the design hierarchy, follow a signal forwards or backwards, use the 'find' input box to search for a signal.

Full desing insight

MAIN WINDOW



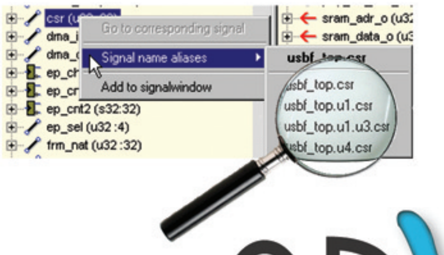
```
161 assign ma_dout = wb_data_i;
162 assign rf_dout = wb_data_i;
163
164 always @(posedge wb_clk)
165     if( `USBF_RF_SEL )  wb_data_o <= rf_din;
166     else                wb_data_o <= ma_din;
167
168 // Sync WISHBONE Request
169 always @(posedge phy_clk)
170     wb_req_s1 <= wb_stb_i & wb_cyc_i;
171
172 // Sync WISHBONE Ack
173 always @(posedge wb_clk)
174     wb_ack_s1 <= wb_ack_d;
175
176 always @(posedge wb_clk)
177     wb_ack_o <= wb_ack_s1 & !wb_ack_s2 & !wb_ack_o;
178
```

The design shown in the window above is the 'USB2.0 Function core' from http:\\www.opencores.com. In the hierarchy the module 'usbf_top.u5' is selected, the right part of the screen shows the input signals, the internal signals, the output signals and the source code of this module. One of the internal signals is 'wb_ack_d'. Icons indicate information about a signal or about signal usage. Signal 'wb_ack_d' is declared as a register and therefore visualized by a flip-flop icon. After the signal container is opened, the usage of the signal is shown. Icons identify left hand or right hand usage in blocking or non-blocking assignment statements. Each time a signal is selected by the mouse the source code pane jumps to the corresponding location in the sources. Here 'wb_ack_d' is read (blue dot) in the left hand side (blue dot is at the left hand side) of a non-blocking assignment statement (<=). The corresponding source code is highlighted on source code line 174.

Items such as: Signals, Wires, Memories, Parameters, Constants, Conditional usage, Edge usage, Clock usage, etc, are recognized and are shown with their corresponding icons. For each occasion of such items a context window is available to jump directly to the appropriate part of the source code. If this source code is part of another module then all panes are updated to visualize that module (focus change).

FEATURES

Context window:
Provides a list of signals in the design hierarchy which equal the selected signal. The signal can be added directly to the Signals waveform.



Undo and Redo button:
Trace the analysis backwards and forwards.

Find box:
Wildcards based (single or multiple) search for signals in total design.

ADVEDA

VERIFICATION SOLUTIONS FOR CHIP DESIGN