

Identify[®]

Simulator-like Visibility into Hardware Debug

Overview

The Identify product is a powerful FPGA verification tool that allows you to quickly find and correct functional design errors in hardware at system speed. The Identify software offers advanced triggering capability so you can focus precisely on the design portion you wish to view, at the time you choose to see it. Most importantly, there's no additional effort required to interpret the results. You add the probes to instrument the design and observe the results directly in your RTL source code.

Debug Where you Design – in RTL Source

The Identify solution lets you debug your hardware like you do in simulation – by controlling hardware triggers and annotating captured data directly in the RTL source code. You can also view the results in a waveform display. Debugger controls let you zoom and pan the waveform at the touch of a button.

RTL Debug Offers Superior Verification

Using simulation, you can perform fast design iterations and gain a comprehensive view of your design. However, this entails long run times and is limited in stimulus to the test vectors you write. The vectors are never more than an approximation of the FPGA operating environment. Test equipment like Logic Analyzers and Mixed Signal Oscilloscopes can probe can probe nodes in designs at system speed and in a real operating environment, but analyzers are pin-limited in the number of design signals they can display. Moreover, reassigning pin-node connectivity requires time-consuming reimplementing of the entire design. The analyzer results then have to be traced back to the source RTL.

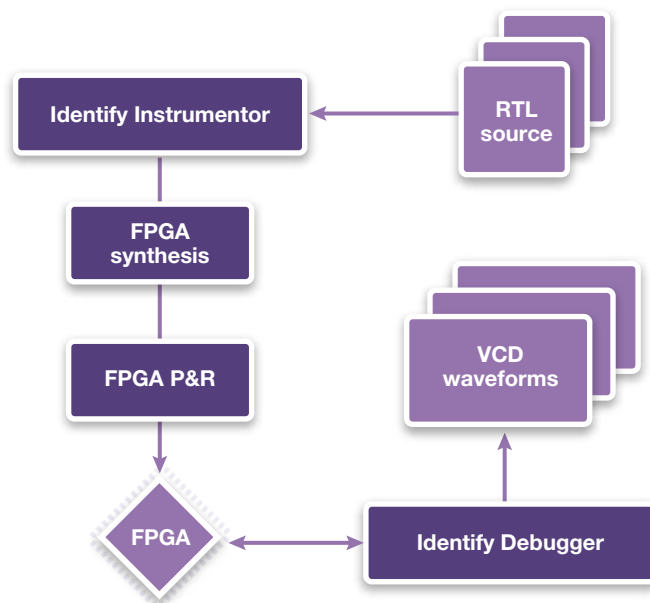


Figure 1: The Identify product supports most popular FPGA devices from Actel, Altera, and Xilinx.

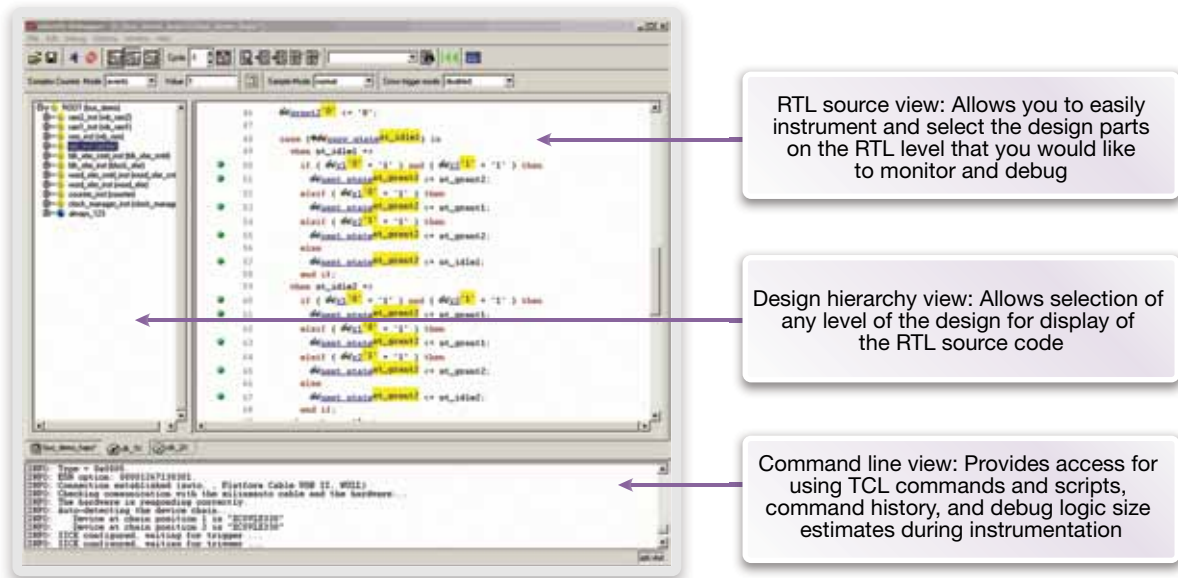


Figure 2: The powerful Identify Debugger GUI allows you to quickly navigate through your design and to debug the RTL source code.

The Identify software offers the speed and accuracy of results achieved with an analyzer along with the breadth and ease of probing of a simulator. There's no limit to the number of signals you can view or the clock domains they are in, and results are obtained as fast as the device can operate.

Trigger Conditions or Code

The Identify RTL Debugger (figure 2), enables you to set triggers on anything from simple signal values or code branches, such as CASE and IF statements, to complex combinations of signals and branches.

You can also create complex sequential state machines to use as precision trigger controls. Adding probes does not affect timing performance and may only increase design size on an average of less than 4% on a small number of devices.

Ease of Use

The Identify product's non-intrusive approach to debug allows you to instrument your design without any changes to your RTL code.

The hierarchy display window lets you navigate quickly to the design module you need, with the compiler automatically placing icons next to nodes designated for probing. You simply activate probes in your Verilog, VHDL, SystemVerilog, or mixed-language design by using menus. For nodes that are triggers, you enter the trigger condition using a pop-up window. You can also assign the debug clock using menus. The Identify solution supports multiple clock domain triggering and clock cross triggering for inter-domain debug.

Once the FPGA is programmed, the Identify Debugger is used to communicate with the FPGA via JTAG interface to set trigger modes and view captured data. Identify Debugger trigger

operating modes include cycles, events, pulse width, and watchdog (figure 3). Use the modes to add clock delays and pulse widths to logic or branch triggers.

Customize your Results

You can use buttons to zoom or pan waveform views to reposition data. A convenient wheel lets you single-step back and forth in design cycle time to observe changes brought about by specific logic.

Debug Iterations Without Place and Route

The Identify solution lets you debug incrementally without re-running place and route. You can move probes dynamically and immediately between nodes to see new data. These advanced features drastically increase productivity.

```

69  next state st_grant2 <= st_idle2;
70  end if;
71  when st_grant1 =>
72  grant1'0' <= '1';

```

Figure 3: Annotated debug values.

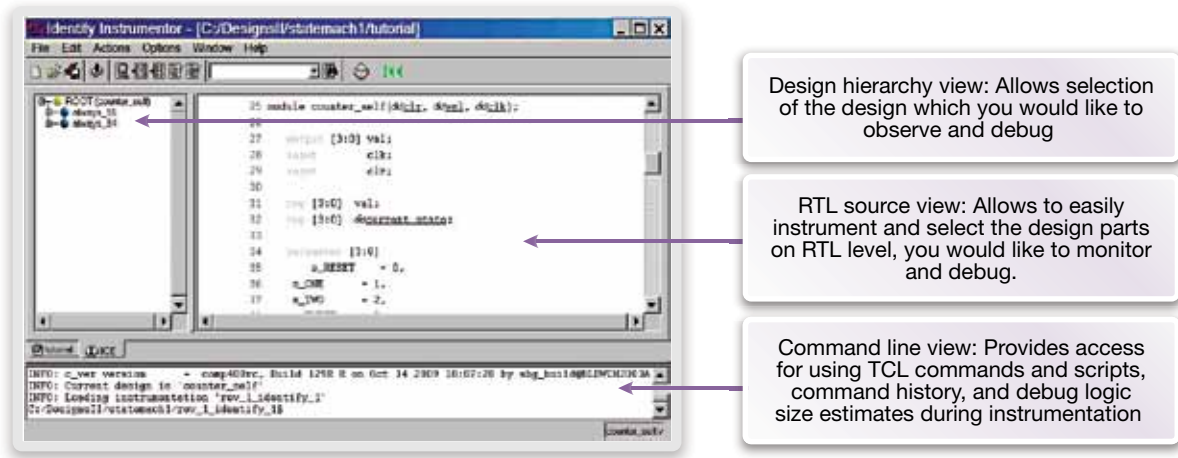


Figure 4: The convenient Identify Instrumentor allows you to easily instrument and select the design parts you would like to observe.

Identify Features	Benefits
Instrument design in RTL source code	Ability to select signals and code branches for sampling and/or triggering easily and quickly
Debug design in RTL source code	Rapid debug of results and the ability to get useful data with less debug logic for heavily utilized FPGAs
Implement triggers on state machines	Useful for creating complex triggering conditions
Qualified sampling	Allows tracking the operation of the design over a longer period of time
Pre-synthesis instrumentation	Combinatorial logic output that needs to be probed is hard to find after synthesis
Supports Verilog, VHDL, SystemVerilog or mixed language	Full support of all HDL design types
Enumerated data type preservation	Displays data in RTL source as symbolic data rather than bit-level, ideal for state machines
Waveform display	Displays captured data in waveform viewers for viewing in a time-based format
Exports debug vectors	Use in-system vectors for simulation
Choose signals for sampling and/or triggering	Minimize debug logic to get just the data you need
No restrictions on signal width and depth	Capture data from any number of signals or depth depending on available area within FPGA
Unlimited sequential trigger conditions	Allows any series of events to be used as a capture trigger
Cross triggering	Triggers from one clock domain which in turn can trigger and sample in another clock domain
Uses built-in or user selected JTAG	Use the built-in JTAG ports and/or user selected pins for JTAG interface
Pipelined debug logic	Minimal or no timing impact on original design
Area estimates during instrumentation	Provides immediate feedback on area used for debug logic
TCL-based command line interface	Allows automation of instrumentation or debug via scripts
Multi-chip support on HAPS	Allowing instrumented sampling logic and design partitioned with Certify® on multiple FPGAs
External trigger Import	Trigger from an external source can be imported and configured as a trigger condition