

Identify[®] Actel Edition Quick Tutorial

September 2010

<http://solvnet.synopsys.com>

SYNOPSYS[®]

Disclaimer of Warranty

Synopsys, Inc. makes no representations or warranties, either expressed or implied, by or with respect to anything in this manual, and shall not be liable for any implied warranties of merchantability or fitness for a particular purpose of for any indirect, special or consequential damages.

Copyright Notice

Copyright © 2010 Synopsys, Inc. All Rights Reserved.

Synopsys software products contain certain confidential information of Synopsys, Inc. Use of this copyright notice is precautionary and does not imply publication or disclosure. No part of this publication may be reproduced, transmitted, transcribed, stored in a retrieval system, or translated into any language in any form by any means without the prior written permission of Synopsys, Inc. While every precaution has been taken in the preparation of this book, Synopsys, Inc. assumes no responsibility for errors or omissions. This publication and the features described herein are subject to change without notice.

Trademarks

Registered Trademarks (®)

Synopsys, AMPS, Astro, Behavior Extracting Synthesis Technology, Cadabra, CATS, Certify, CHIPit, CoMET, Design Compiler, DesignWare, Formality, Galaxy Custom Designer, HAPS, HapsTrak, HDL Analyst, HSIM, HSPICE, Identify, Leda, MAST, METeor, ModelTools, NanoSim, OpenVera, PathMill, Physical Compiler, PrimeTime, SCOPE, Simply Better Results, SiVL, SNUG, SolvNet, Syndicated, Synplicity, the Synplicity logo, Synplify, Synplify Pro, Synthesis Constraints Optimization Environment, TetraMAX, UMRBus, VCS, Vera, and YIELDirector are registered trademarks of Synopsys, Inc.

Trademarks (™)

AFGen, Apollo, Astro-Rail, Astro-Xtalk, Aurora, AvanWaves, BEST, Columbia, Columbia-CE, Confirma, Cosmos, CosmosLE, CosmosScope, CRITIC, DC Expert, DC Professional, DC Ultra, Design Analyzer, Design Vision, Design-erHDL, DesignPower, Direct Silicon Access, Discovery, Eclipse, Encore,

EPIC, Galaxy, HANEX, HAPS, HapsTrak, HDL Compiler, Hercules, Hierarchical Optimization Technology, High-performance ASIC Prototyping System, HSIM, HSIM^{plus}, i-Virtual Stepper, IICE, in-Sync, iN-Tandem, Jupiter, Jupiter-DP, JupiterXT, JupiterXT-ASIC, Liberty, Libra-Passport, Library Compiler, Magellan, Mars, Mars-Rail, Mars-Xtalk, Milkyway, ModelSource, Module Compiler, MultiPoint, Physical Analyst, Planet, Planet-PL, Polaris, Power Compiler, Raphael, Saturn, Scirocco, Scirocco-i, Star-RCXT, Star-SimXT, System Compiler, System Designer, Taurus, TotalRecall, TSUPREM-4, VCS Express, VCSi, VHDL Compiler, VirSim, and VMC are trademarks of Synopsys, Inc.

Service Marks (SM)

MAP-in, SVP Café, and TAP-in are service marks of Synopsys, Inc.

SystemC is a trademark of the Open SystemC Initiative and is used under license. ARM and AMBA are registered trademarks of ARM Limited. Saber is a registered trademark of SabreMark Limited Partnership and is used under license. All other product or company names may be trademarks of their respective owners.

Restricted Rights Legend

Government Users: Use, reproduction, release, modification, or disclosure of this commercial computer software, or of any related documentation of any kind, is restricted in accordance with FAR 12.212 and DFARS 227.7202, and further restricted by the Synopsys Software License and Maintenance Agreement. Synopsys, Inc., Synplicity Business Group, 700 East Middlefield Road, Mountain View, CA 94043, U. S. A.

Printed in the U.S.A
September 2010



Introduction

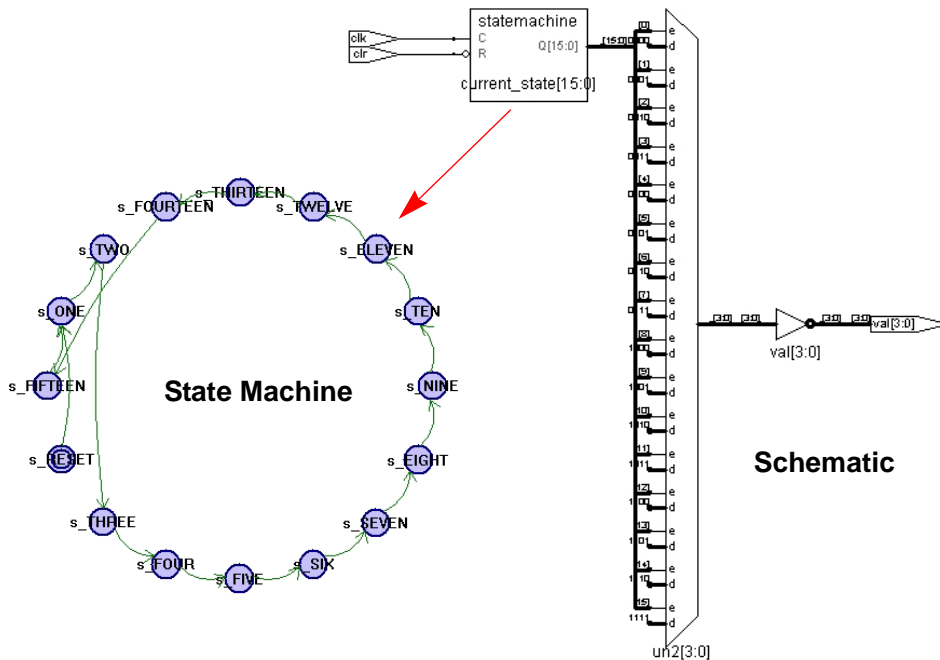
This simple tutorial teaches you how to instrument and debug a small HDL design. The design is a simple 4-bit counter with a clock and reset. Two versions of the counter are provided: one in VHDL and one in Verilog.

Note: This tutorial simulates hardware debug data by applying randomly generated data to all instrumented nodes. This data does not reflect the actual operation of the design and only serves to show the format of the debug data.

Note: A more detailed tutorial, which can be used with actual hardware, is included in the documentation set.

Design Schematic

The following figure shows the simple state machine configured as a 4-bit counter. The state diagram is shown to the left of the schematic.



Design Description

The tutorial design is implemented in Verilog as a single module with two always block statements and in VHDL as a single entity with two processes. The first always block (Verilog) or process (VHDL) implements a state machine; the second always block or process computes the output values based on the current state.

Instrumenting Your Design



You use the Identify instrumentor to select both breakpoints and watchpoints and to set the sampling and triggering modes. The Identify instrumentor is launched from your Synplify Pro synthesis tool and is run prior to synthesis.

Note: This tutorial describes running the Identify instrumentor from the Synplify Pro tool. To run the Identify instrumentor in stand-alone mode, see Chapter 3, *Project Handling*, in the user guide.

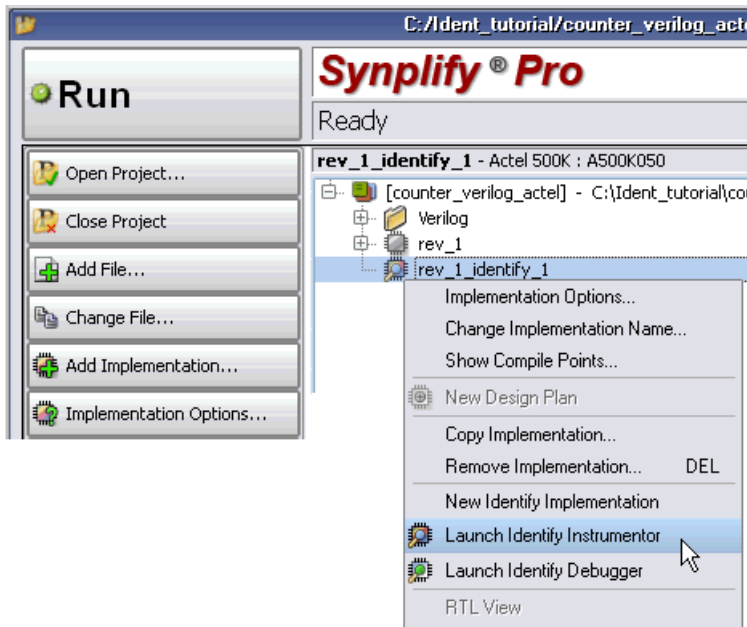
The HDL design and project files for this tutorial are included in a “tutorial” subdirectory under the Identify software installation directory. Before you begin the tutorial, copy the files to a local directory and make sure that you have read and write permission for both the directory and files.

Note: While performing the tutorial, the active project (`.prj`) file will be updated; copying the files to a local directory preserves the original files installed in the tutorial directory.

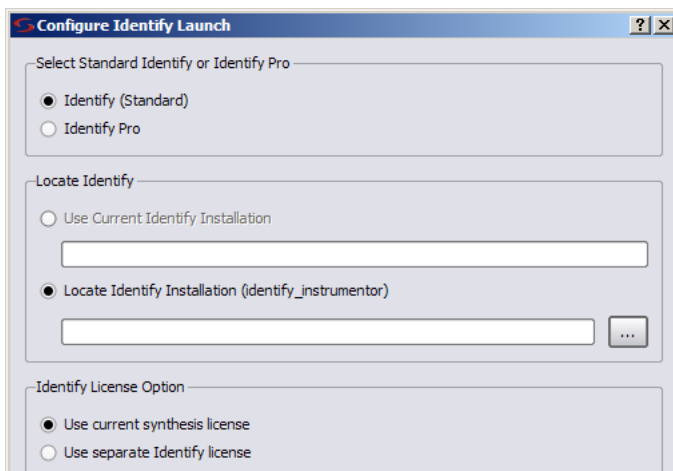
To begin the instrumentation:

1. Start the Synplify Pro tool.
2. In the project view, click the Open Project button to display the Open Project dialog box and click the Existing Project button.
3. Navigate to the tutorial directory where the Identify software is installed. This directory includes the HDL design files and two Actel-specific project files for Verilog and VHDL implementations.
4. Select (open) the appropriate project file.

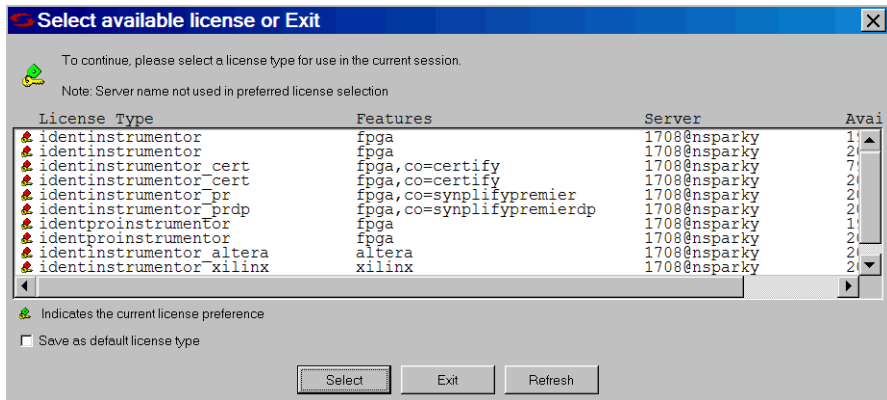
5. Right click on the Identify implementation and select Launch Identify Instrumentor from the popup menu.



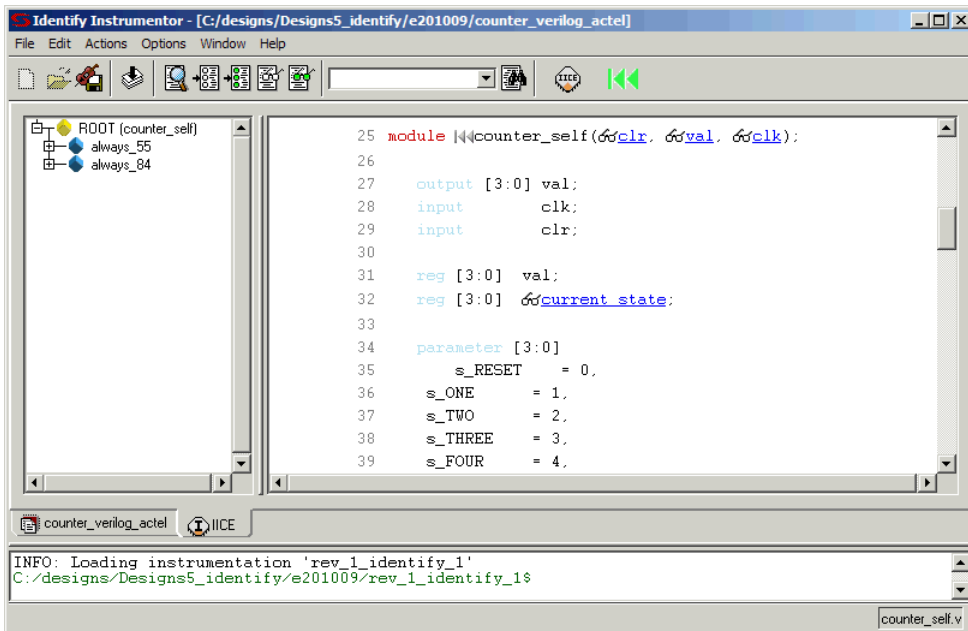
6. If prompted, enter the location of the Identify installation in the Configure Identify Launch dialog box, click the Locate Identify Installation radio button, and click OK to launch the Identify instrumentor.



7. If prompted for a license, select a license from the list of available licenses displayed and click Select.



The following figure shows the initial Identify instrumentor window as launched from the Synplify Pro tool on the Verilog version of the tutorial. The window shows the design hierarchy on the left and the HDL file content with all the potential instrumentation marked and available for selection on the right.



Setting up the IICE

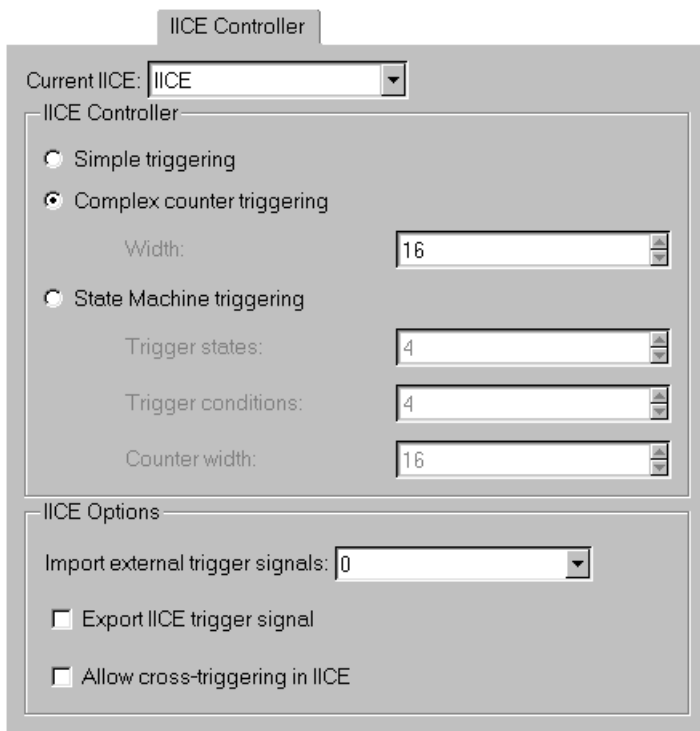


Click on the Edit IICE settings icon on the toolbar to bring up the IICE Sampler tab shown in the following figure. The IICE Sampler tab defines the sample depth, sampling modes, and the sample clock.

The IICE Sampler dialog box is a rectangular window with a title bar that says 'IICE Sampler'. It contains several settings sections. The 'Current IICE' section has a dropdown menu set to 'IICE'. The 'IICE Sampler' section includes a 'Buffer type' dropdown set to 'behavioral', a 'Sample depth' spinner set to '128', and two unchecked checkboxes: 'Allow qualified sampling' and 'Allow always-armed triggering'. The 'Sample Clock' section includes a 'Sample clock' text field containing '/clk' and a 'Clock edge' section with two radio buttons: 'Positive' (which is selected) and 'Negative'.

1. Leave Buffer type set to behavioral (only supported type)
2. Select 128 for the sample buffer depth.
3. Leave the Allow qualified sampling check box unchecked
4. Leave the Allow always-armed sampling check box unchecked
5. Enter /clk for the sample clock and select the positive polarity for the clock edge.
6. After you have set and/or verified the above IICE Sampler tab settings, click the IICE Controller tab.

The IICE Controller tab selects the type of triggering.

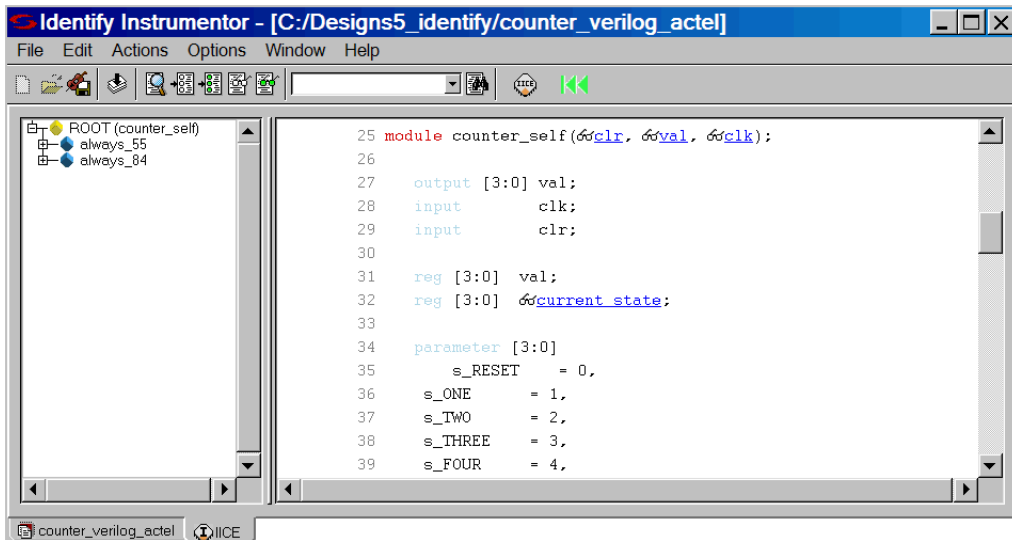


On the IICE Controller tab:

1. Make sure that the Complex counter triggering radio button is selected and that the Width is set to 16.
2. Leave the Import external trigger signals value at 0.
3. Leave the Export IICE trigger signal and Allow cross-triggering in IICE check boxes unchecked.
4. Click the OK button at the bottom of the dialog box.

Selecting the Instrumentation

After setting up the IICE, the HDL code for the tutorial design is displayed in the Identify instrumentor window as shown in the following figure. Use the hierarchy browser on the left to navigate through your design. Clicking on a hierarchical node displays the corresponding source code.



Selecting Watch Points



In the source code display, scroll down and select the signal `current_state` on line 32 (Verilog) or line 52 (VHDL) for instrumentation by clicking on the watch-point (glasses) icon displayed next to its name. When you click on the icon (or on the signal name), a popup menu is displayed as shown in the following figure to allow you to select how the watch-point signal is to be instrumented.

```

52   signal  current_state: state;
53   begin
54
55   process( clk, 
56   begin
57     if  clr = '0' then
58        current_state <= s_RESET;
59     elsif  clk'event and  clk = '1' then
60       case  current_state is
61         when s_RESET    =>  current_state <= s_ONE;
62         when s_ONE      =>  current_state <= s_TWO;
63         when s_TWO      =>  current_state <= s_THREE;
64         when s_THREE    =>  current_state <= s_FOUR;

```

Select **Sample and trigger** for the `current_state` signal. The icons preceding each occurrence of the signal in the HDL code will be green and an estimate of the hardware overhead required to instrument the signal will be displayed in the console window.

Note that when you select an instrumentation type, the icon changes color according to the following table.

Icon Color	Watch-Point Selection
Green	Sample and trigger
Blue	Sample only
Pink	Trigger only
Clear (unfilled)	Not instrumented

Selecting Breakpoints

The icons to the left of the line numbers beginning on line 61 select the corresponding breakpoint for instrumentation. When selected, the color of the icon changes to green. Click on the icons on lines 62, 68, and 69 to select their corresponding breakpoints.

```

58     current_state <= s_RESET;
59     else if clk'event and clk = '1' then
60         case current_state is
61             when s_RESET    => current_state <= s_ONE;
62             when s_ONE      => current_state <= s_TWO;
63             when s_TWO      => current_state <= s_THREE;
64             when s_THREE    => current_state <= s_FOUR;
65             when s_FOUR     => current_state <= s_FIVE;
66             when s_FIVE     => current_state <= s_SIX;
67             when s_SIX      => current_state <= s_SEVEN;
68             when s_SEVEN    => current_state <= s_EIGHT;
69             when s_EIGHT    => current_state <= s_NINE;
70             when s_NINE     => current_state <= s_TEN;

```

Writing the Instrumented Design



To write the instrumented design, select File->Save project from the menu or click on the Save and Instrument current project icon on the toolbar. Saving the project automatically updates the subdirectory `rev_1_identify_1` subdirectory with the instrumented HDL code for the tutorial design and adds the Identify project file `counter_verilog_actel.bsp` or `counter_vhdl_actel.bsp` to the top-level directory where it is read by the Identify debugger.

Synthesizing Your Design

The Identify tool set is expressly designed to work with the Synopsys Synplify Pro synthesis tool. After instrumenting and saving your design, the instrumented design is synthesized in the Synplify Pro tool as described in the FPGA Synthesis tool documentation.

Place and Route

After synthesis, run the file generated by your Synplify Pro tool through your place and route tool. Running the file of the instrumented design through place and route is identical to running place and route on the original design. Refer to your place and route tool documentation for further information.

If you have selected the soft JTAG port setting, four external ports have been added to the tutorial design. These ports are:

- identify_jtag_tdi (input serial data IN signal.)
- identify_jtag_tck (input asynchronous clock signal)
- identify_jtag_tms (input control signal)
- identify_jtag_tdo (output serial data OUT signal.)

You must provide proper pin locations for these ports to your place and route tools. These pins must be connected to your JTAG cable to enable communication with the Identify debugger. Please refer to the user guide for more information about this process.

Program the Device

Programming the target device with the bit file of the instrumented design is identical to programming the target device with the original design's bit file. Please refer to your programming tool documentation for further information.

Debugging Your Design

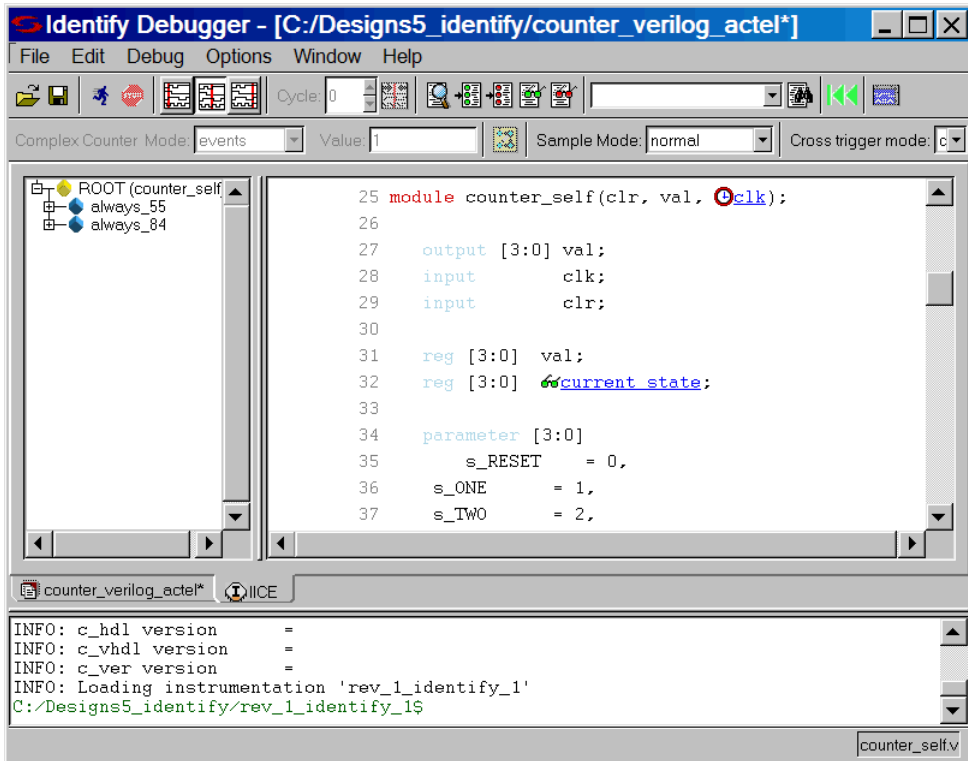
Debugging your design is done from the Identify debugger. To launch the debugger from your Synplify Pro synthesis tool:

1. Open the project and highlight the Identify implementation in the Project view.
2. With the right mouse button, select Launch Identify Debugger from the popup menu or click the Launch Identify Debugger icon in the top menu bar.

If you are prompted for a license, select the appropriate license from the list of available licenses displayed.

Note: To avoid being prompted for a license each time you start the Identify debugger, check the Save as default license type box before selecting your license.

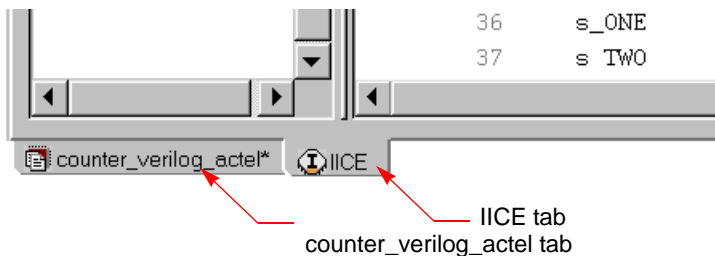
The Identify debugger window opens your project in the instrumentation window with the hierarchy browser displayed on the left and the HDL source code displayed on the right as shown in the following figure. Note that the only instrumentations visible in the source code display are the breakpoints and watchpoints that you selected during the instrumentation phase.



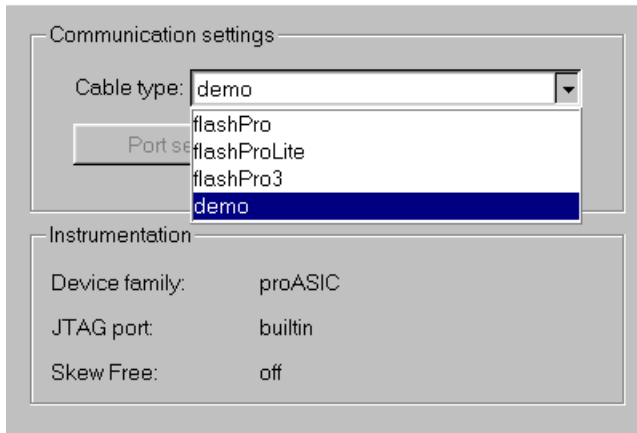
Selecting the Cable Type

To run the tutorial, select the “demo” cable type by:

1. Clicking on the “counter” tab at the lower left corner of the window to display the project window.



2. Selecting demo from the Cable type drop down menu.



3. Clicking on the IICE tab to redisplay the instrumentation window.

Triggering on a Breakpoint

In the source code display, use the scroll bar to scroll down until the first breakpoint on line 62 is visible on the left side of the source code and then click on the breakpoint to activate it.

```

58     current_state = s_RESET; /* 4'b0000 */
59     else begin
60         case (current_state)
61     s_RESET: current_state = s_ONE;
62     s_ONE: current_state = s_TWO;
63     s_TWO: current_state = s_THREE;
64     s_THREE: current_state = s_FOUR;
65     s_FOUR: current_state = s_FIVE;
66     s_FIVE: current_state = s_SIX;
67     s_SIX: current_state = s_SEVEN;
68     s_SEVEN: current_state = s_EIGHT;
69     s_EIGHT: current_state = s_NINE;
70     s_NINE: current_state = s_TEN;

```

Notice that the breakpoint icon changes from green to red indicating that the breakpoint is active. The breakpoint at line 62 triggers on the positive edge of the sample clock when the `current_state` signal has the value `s_ONE`.



Now that you have an active trigger condition, arm the IICE trigger circuits on the FPGA device by clicking the Run (Arm selected IICE(s) for triggering) icon in the menu bar.

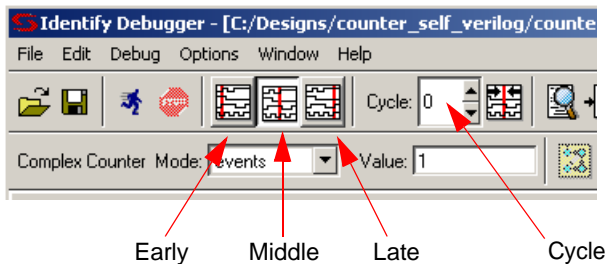
Clicking on the Run icon downloads the trigger information to the IICE. When the trigger occurs, the sampled data is transferred back to the debugger. The small arrow to the left of the breakpoint icon indicates which breakpoint triggered (identifying which breakpoint triggered is important when multiple breakpoints are active).

```

58     current_state4'b0110 = s_RESET; /* 4'b0000
59     else begin
60         case (current_state4'b0110)
61     s_RESET: current_state4'b0110 = s_ONE;
62     s_ONE: current_state4'b0110 = s_TWO;
63     s_TWO: current_state4'b0110 = s_THREE;
64     s_THREE: current_state4'b0110 = s_FOUR;
65     s_FOUR: current_state4'b0110 = s_FIVE;
66     s_FIVE: current_state4'b0110 = s_SIX;
67     s_SIX: current_state4'b0110 = s_SEVEN;
68     s_SEVEN: current_state4'b0110 = s_EIGHT;
69     s_EIGHT: current_state4'b0110 = s_NINE;

```

The Cycle display in the middle of the menu bar shows the value zero where the trigger occurred. By clicking on the up-down arrows on the right, you can increase or decrease the cycle count to show values before or after the trigger point.



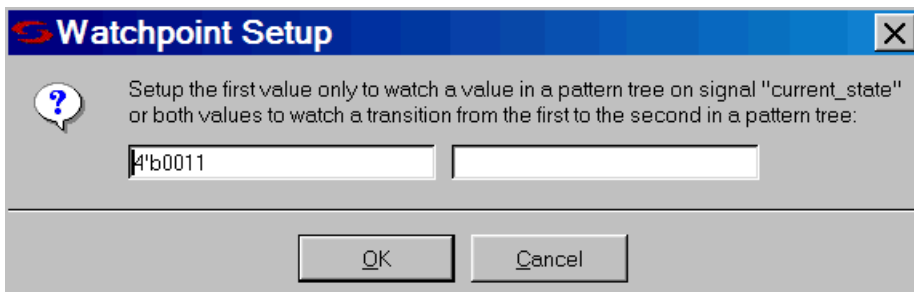
You can change where the trigger point is in the buffer by selecting one of the Early, Middle, or Late icons to the left of the cycle counter and again clicking the Run icon. The trigger location changes the next time the IICE triggers.

Triggering on a Watchpoint

You can also trigger on a watchpoint that is specified on any sampled signal. The Watchpoint Setup dialog box accepts any legal VHDL or Verilog expression that evaluates to a constant.

To set a simple watchpoint:

1. Click on the `current_state` signal
2. Select Set trigger expressions from the popup menu
3. In the First value field, enter `s_THREE` (VHDL) or `4'b0011` (Verilog)
4. Click OK





Click the Run icon. When signal `current_state` reaches the value `s_THREE` (VHDL) or `4'b0011` (Verilog), the IICE triggers.

Note: Because randomly generated data is applied, the trigger watchpoint (`s_THREE` or `4'b0011`) may not reach its intended value. Click the adjacent STOP icon if triggering does not occur within a few seconds.

Using the Cycle data display controller, you can now browse back and forth through the debugger data buffer to view the design activity.

Cycle data display controller

```

58     current_state4'b0011 = s_RESET; /* 4'b0000 */
59     else begin
60         case (current_state4'b0011)
61     s_RESET: current_state4'b0011 = s_ONE;
62     s_ONE: current_state4'b0011 = s_TWO;
63     s_TWO: current_state4'b0011 = s_THREE;
64     s_THREE: current_state4'b0011 = s_FOUR;
65     s_FOUR: current_state4'b0011 = s_FIVE;
66     s_FIVE: current_state4'b0011 = s_SIX;
67     s_SIX: current_state4'b0011 = s_SEVEN;
68     s_SEVEN: current_state4'b0011 = s_EIGHT;
    
```

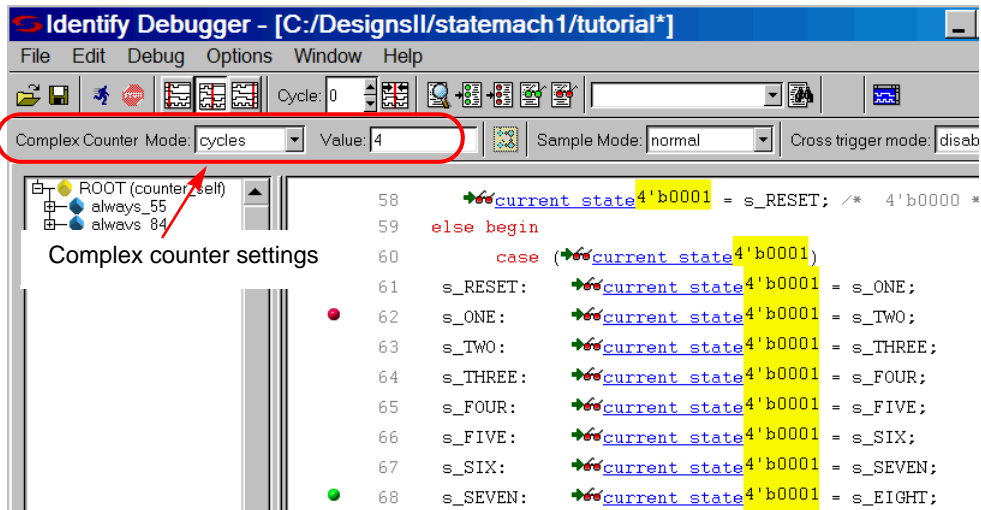
Using the Complex Counter

The default settings for the complex counter mode (events with a value of 1) effectively disable the counter. To use the complex counter to wait for a breakpoint and/or watchpoint trigger event and then to count a specified number of cycles before triggering the sample buffer:

1. Set Complex Counter Mode to cycles and Value to a value greater than 1.
2. Change the watchpoint of signal `current_state` to `s_ONE` (VHDL) or `4'b0001` (Verilog).
3. Click the Run icon and wait for the data to download.

The value at time zero will be updated with the sample data after the specified number of cycles have occurred as shown in the following figure.

Note: Because randomly generated data is applied to all instrumented nodes, the results displayed will not reflect actual design operation.

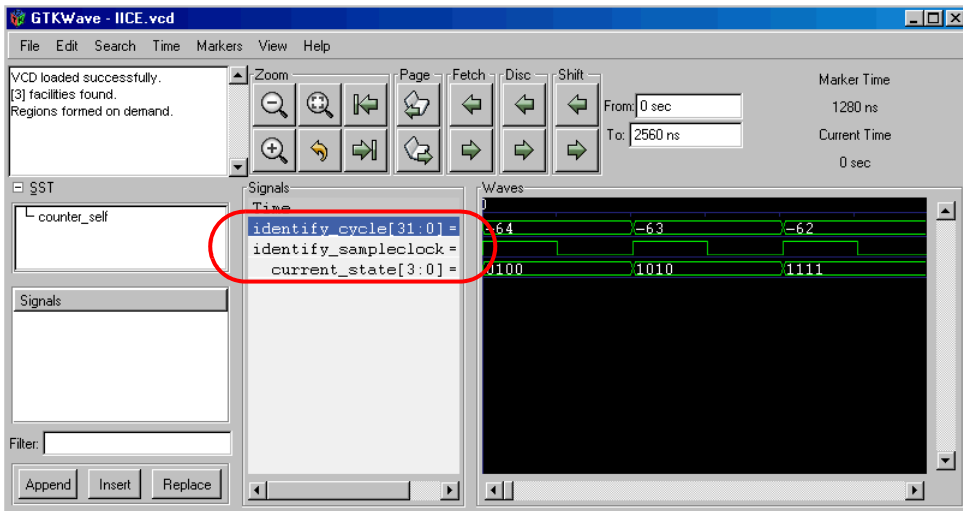


Generating Waveforms



Display the debug data by clicking the Open Waveform Display icon in the menu bar.

All sampled signals are included in the waveform display with two additional signals automatically added at the top of the display. The first signal, `identify_cycle`, shows the trigger location in the sample buffer. The second signal, `identify_sampleclock`, shows every clock edge. The following figure shows a typical waveform view with the `identify_cycle` and `identify_sampleclock` signals highlighted.





Synopsys, Inc.

700 East Middlefield Road
Mountain View, CA 94043 USA
www.solvnet.com

Copyright © 2010 Synopsys, Inc. All rights reserved. Specifications subject to change without notice. Synopsys, Behavior Extracting Synthesis Technology, Certify, DesignWare, HDL Analyst, Identify, SCOPE, "Simply Better Results", SolvNet, Synplicity, the Synplicity logo, Synplify, Synplify ASIC, Synplify Pro, Synthesis Constraints Optimization Environment, and VCS are registered trademarks of Synopsys, Inc. BEST, Confirma, HAPS, HapsTrak, High-performance ASIC Prototyping System, IICE, MultiPoint, Physical Analyst, System Designer, and TotalRecall are trademarks of Synopsys, Inc. All other names mentioned herein are trademarks or registered trademarks of their respective companies.