Identify® Microsemi Edition Quick Tutorial

December 2015

SYNOPSYS®

Copyright Notice and Proprietary Information

Destination Control Statement

All technical data contained in this publication is subject to the export control laws of the United States of America. Disclosure to nationals of other countries contrary to United States law is prohibited. It is the reader's responsibility to determine the applicable regulations and to comply with them.

Disclaimer

SYNOPSYS, INC., AND ITS LICENSORS MAKE NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS MATERIAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

Trademarks

Synopsys and certain Synopsys product names are trademarks of Synopsys, as set forth at
http://www.synopsys.com/Company/Pages/Trademarks.aspx.
All other product or company names may be trademarks of their respective owners.

Third-Party Links

Any links to third-party websites included in this document are for your convenience only. Synopsys does not endorse and is not responsible for such websites and their practices, including privacy practices, availability, and content.

December 2015

Quick Tutorial

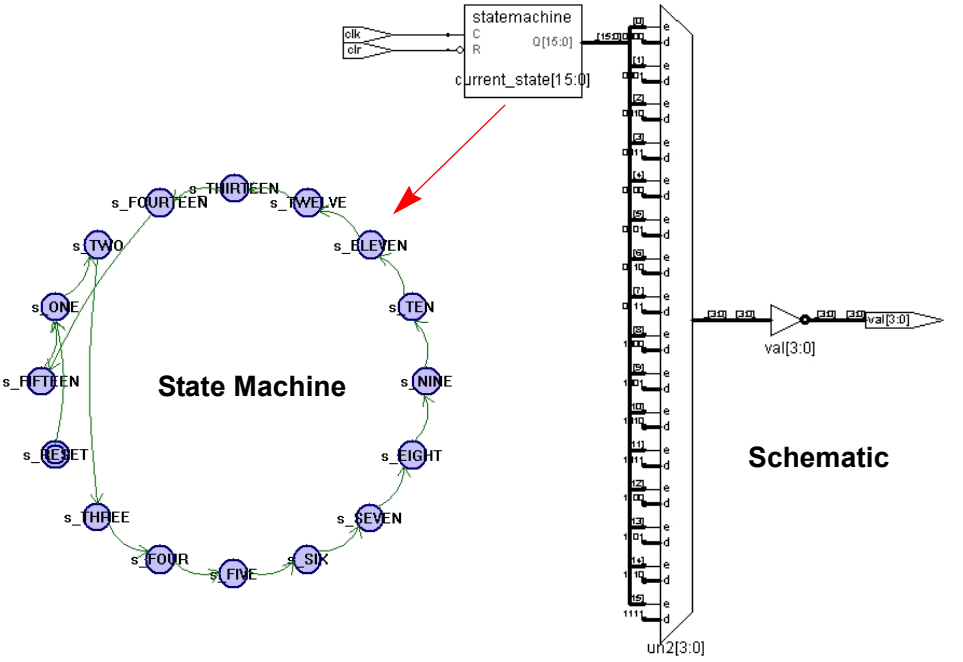
This simple tutorial teaches you how to instrument and debug a small HDL design. The design is a simple 4-bit counter with a clock and reset. The counter design used with the tutorial is written in VHDL.

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.

This tutorial introduces the reader to basic Identify operations. The tutorial includes the following major topics which are intended to be performed in the order listed below.

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 VHDL as a single entity with two processes. The first process implements a state machine; the second 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 the Synplify Pro synthesis tool and is run prior to synthesis.

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

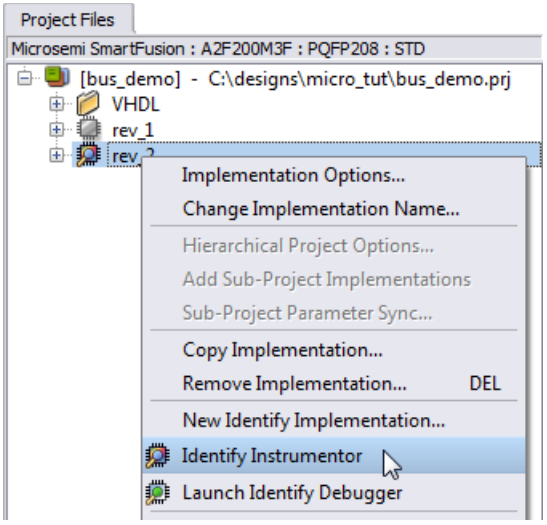
Note: While performing the tutorial, the active project file will be updated; copying the files to a local directory preserves the original files installed in the share directory.

To begin the instrumentation:

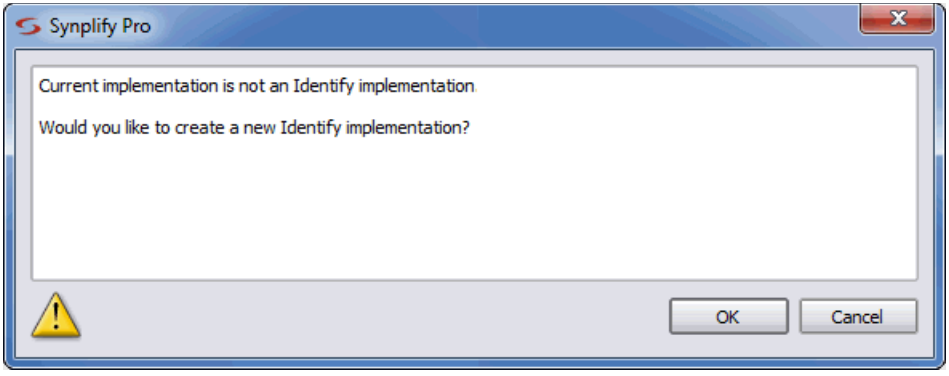
1. Start the Synplify Pro synthesis tool.
2. In the project view, click the Open Project button to display the Open Project dialog box and click the Existing Project button.
3. Navigate to the location where you copied the counter_vhdl subdirectory. This subdirectory includes an HDL design file (counter.vhd) and a Microsemi-specific project file (counter_vhdl.prj).
4. Select (open) the Microsemi-specific project file.
5. Select File->Save As from the menu and rename the selected project file to tutorial.prj.

   Note: The remainder of this document uses the term *tutorial* to reference the VHDL project.

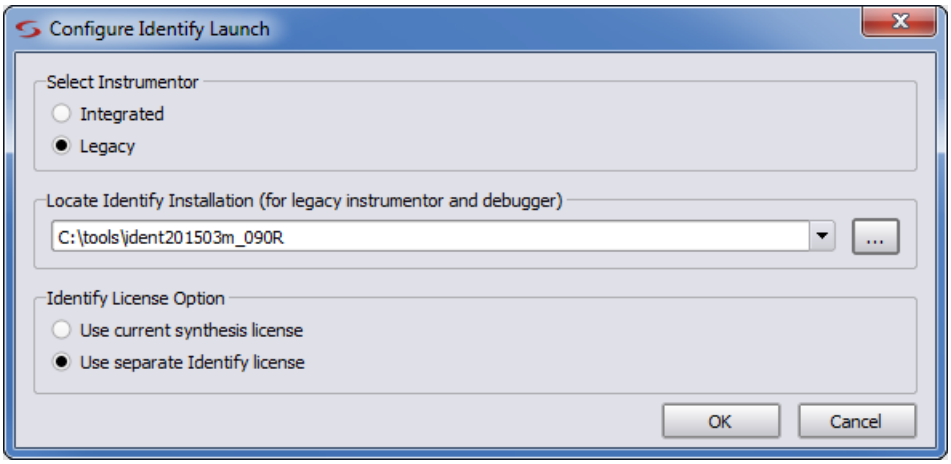
6. Right click on the Identify implementation and select Identify Instrumentor from the popup menu.



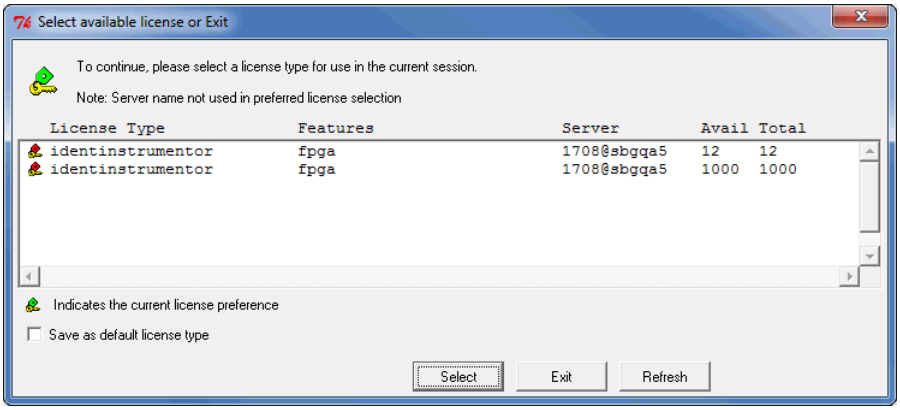
7. If prompted to create a new Identify implementation, click OK.



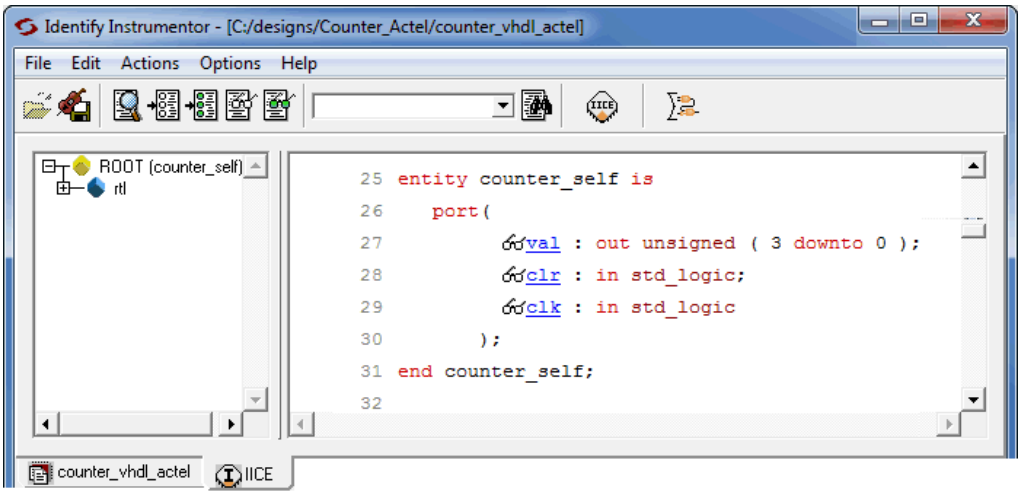
8. If prompted, enter the location of the Identify installation in the Configure Identify Launch dialog box, click the Locate Identify Installation radio button, and click OK to launch the Identify instrumentor.



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

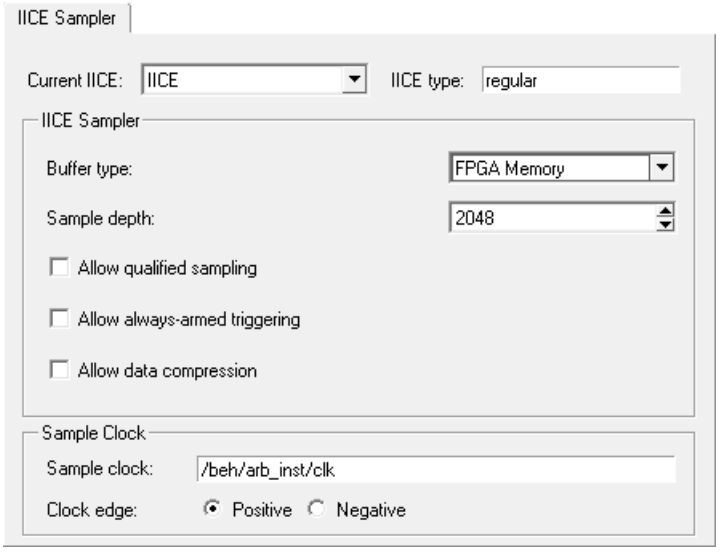


The figure below shows the initial Identify instrumentor window as launched from the Synplify Pro synthesis tool. The window shows the design hierarchy on the left and the HDL file content with all the potential instrumentation marked and available for selection on the right.



Setting up the IICE

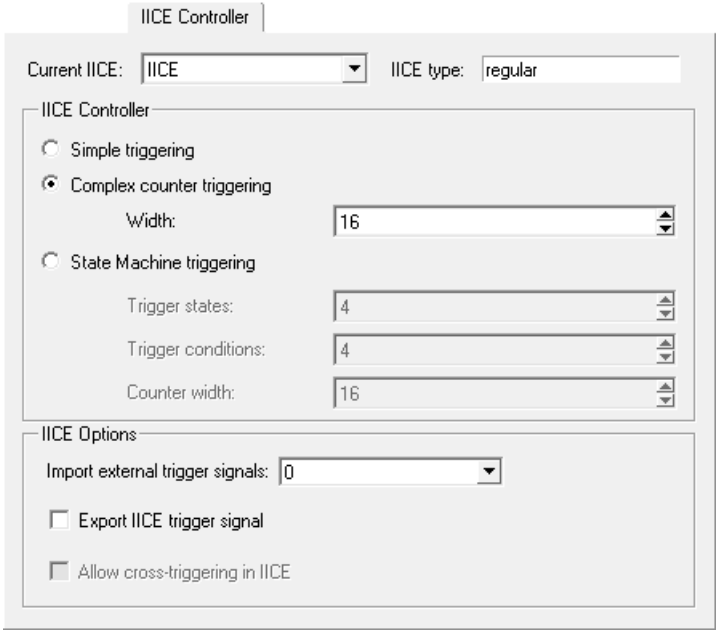
Click on the Edit IICE settings icon on the toolbar to bring up the IICE Sampler tab shown in the following figure. The IICE Sampler tab defines the sample depth, sampling modes, and the sample clock.



On the IICE Sampler tab:

1. Leave Buffer type set to internal_memory
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. Leave the Allow data compression check box unchecked.
6. Enter /clk for the sample clock and select positive polarity for the clock edge.
7. 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.



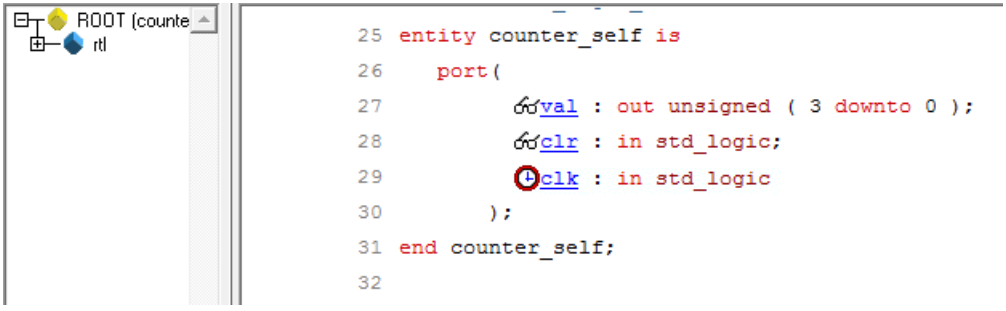
On the IICE Controller tab:

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 set to 0.
3. Leave the Export IICE trigger signal check box unchecked; the Allow cross-triggering in IICE check box cannot be selected until a second IICE unit is created.
4. Click the OK button at the bottom of the dialog box.

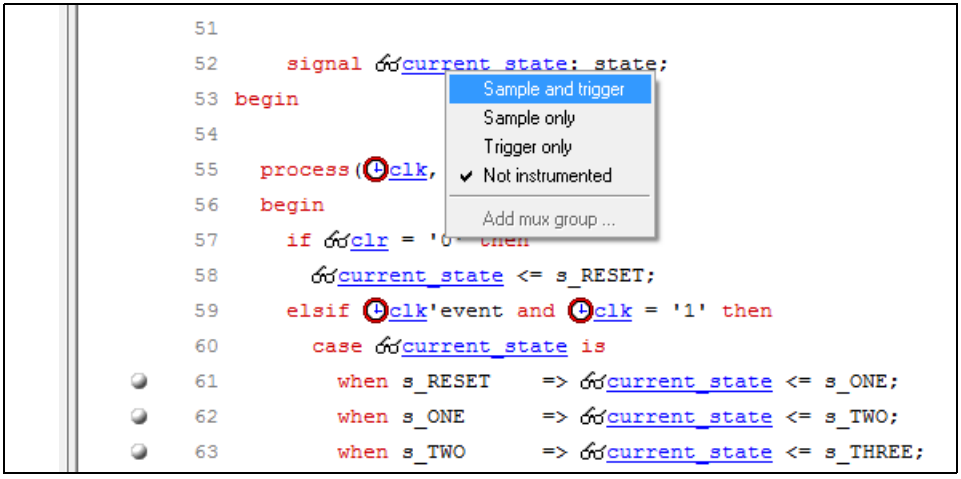
Selecting the Instrumentation

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 section of the source code.



Selecting Watch Points

In the source code display, scroll down and select the signal current_state on line 52 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.



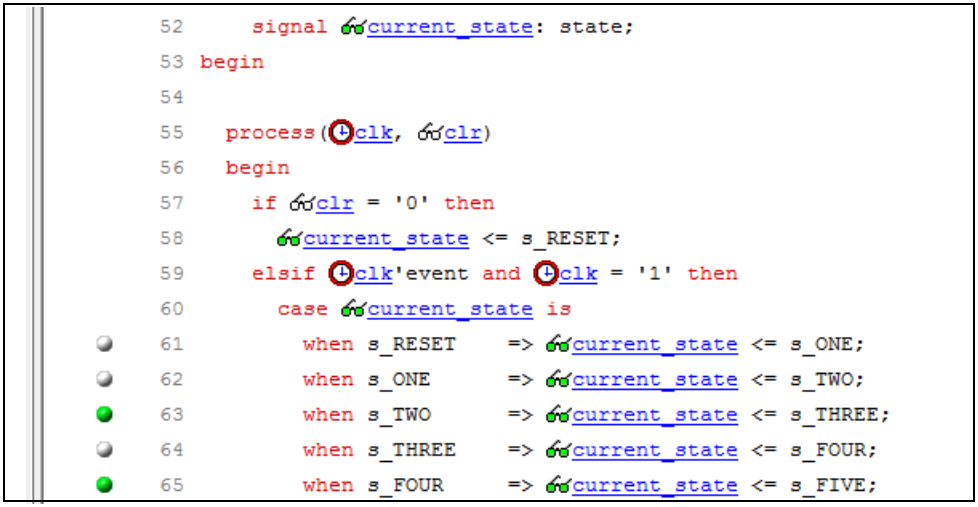
Select Sample and trigger for the current_state signal. The icons preceding each occurrence of the signal in the HDL code will be green and an accumulation of the total number of bits for each instrumentation type will be displayed in the console window.

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 circular 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 63, 65, and 67 to select their corresponding breakpoints.



Writing the Instrumented Design

To write the instrumented design, select File->Save project instrumentation from the menu or click on the Save project's activated instrumentation icon on the toolbar. Saving the project automatically adds a set of files to the Identify implementation directory which are then used by the synthesis tool to incorporate the instrumented logic into the design.

At this point, you would:

- synthesize the design in the Synplify Pro synthesis tool to generate the output netlist
- place and route the synthesized output netlist in the Libero place and route tool
- program the resultant bit file into the FPGA
- cable the board containing the programmed FPGA to your host for analysis by the Identify debugger

Debugging Your Design

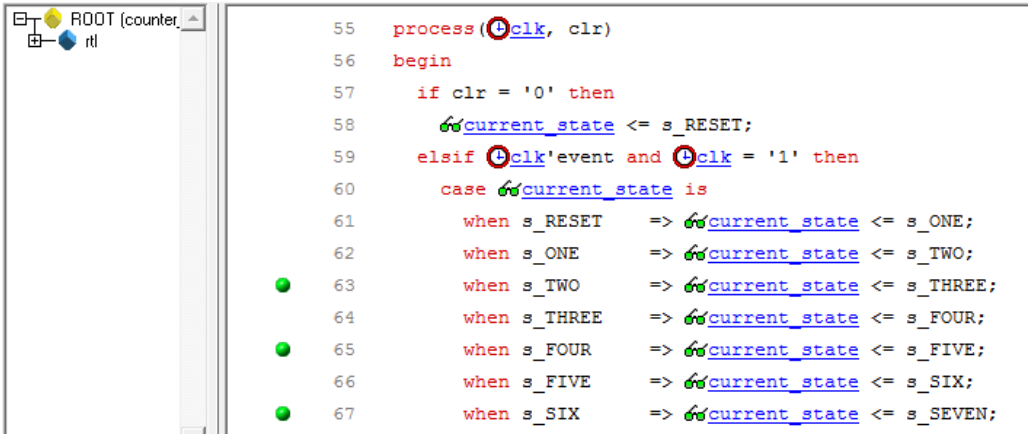
Debugging your design is done from the Identify debugger. To launch the debugger from the Synplify Pro synthesis tool:

1. Open the tutorial project in the synthesis tool 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.

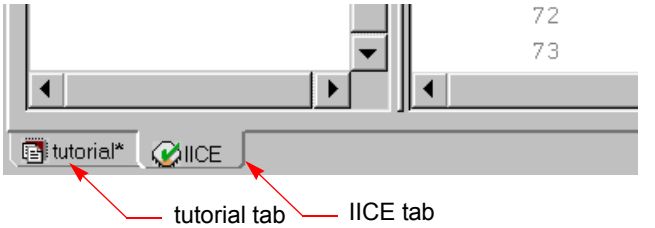
The Identify debugger opens your project in the instrumentation window with the hierarchy browser displayed on the left and the HDL source code displayed on the right as shown in the following figure. Note that the only instrumentation visible in the source code display are the breakpoints and watchpoints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



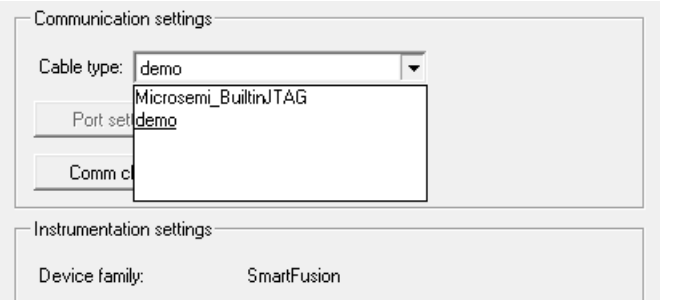
Selecting the Cable Type

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

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



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



If a demo selection is not available, enter the following Tcl command at the console window prompt:

```
com cabletype demo
```

3. Clicking on the IICE tab at the lower left corner of the window 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 63 is visible on the left side of the source code and then click on the breakpoint to activate it.

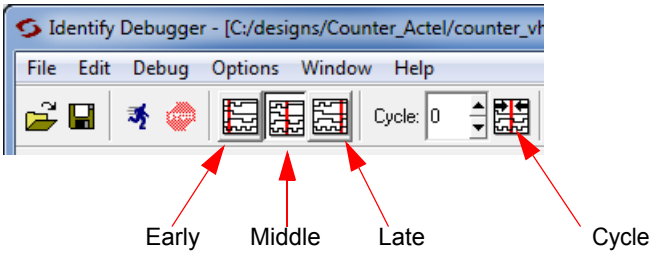
```
55   process(clk, clr)
56   begin
57     if clr = '0' then
58       current_state <= s_RESET;
59     elsif clk'event and clk = '1' then
60       case current_state is
61         when s_RESET    => current_state <= s_ONE;
62         when s_ONE      => current_state <= s_TWO;
63         when s_TWO      => current_state <= s_THREE;
64         when s_THREE    => current_state <= s_FOUR;
65         when s_FOUR     => current_state <= s_FIVE;
```

Notice that the breakpoint icon changes from green to red indicating that the breakpoint is active. The breakpoint at line 63 triggers on the positive edge of the sample clock when the current_state signal has the value s_TWO.

Now that you have an active trigger condition, arm the IICE trigger circuits on the FPGA device by clicking the Run 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).

```
55   process(clk, clr)
56   begin
57     if clr = '0' then
58       current_states_thirteen <= s_RESET;
59     elsif clk'event and clk = '1' then
60       case current_states_thirteen is
61         when s_RESET    => current_states_thirteen <= s_ONE;
62         when s_ONE      => current_states_thirteen <= s_TWO;
63         when s_TWO      => current_states_thirteen <= s_THREE;
64         when s_THREE    => current_states_thirteen <= s_FOUR;
65         when s_FOUR     => current_states_thirteen <= s_FIVE;
```

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 immediately before or after the trigger point.



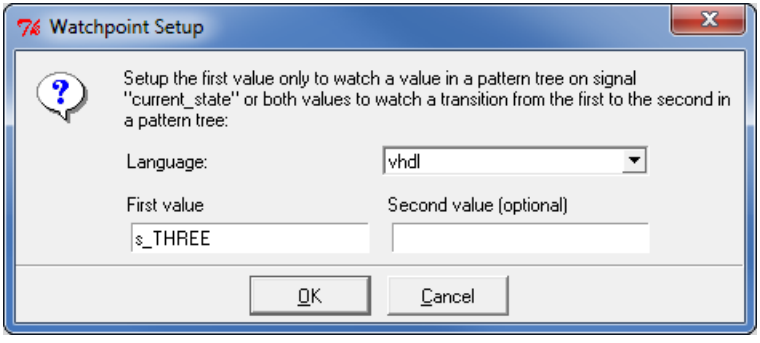
You can change where the trigger point is in the buffer by selecting one of the Early, Middle, or Late icons to the left of the cycle counter and again clicking the Run icon. The trigger location changes the next time the IICE triggers.

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:

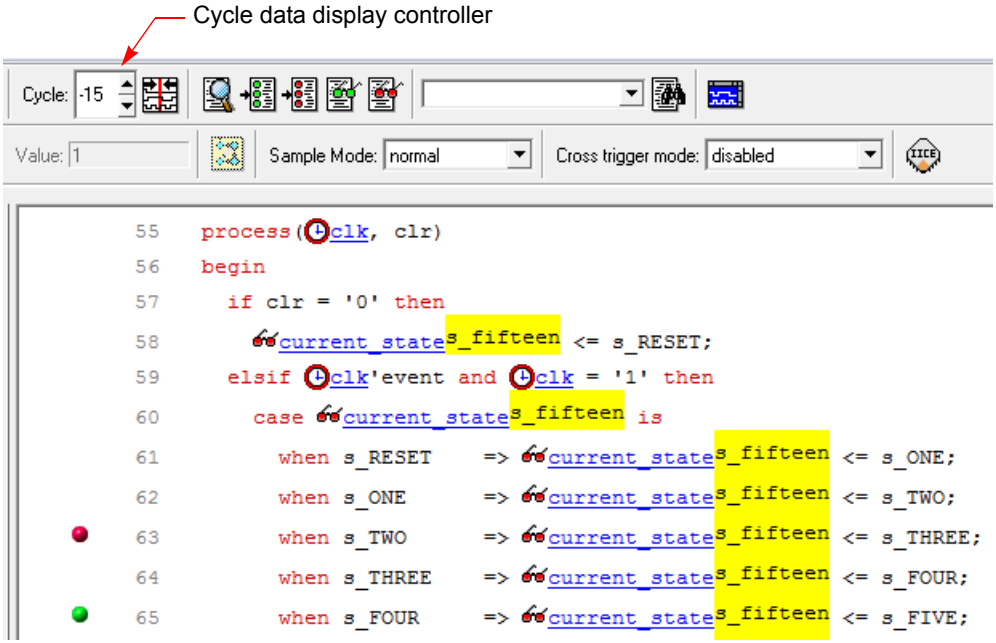
1. Click on the current_state signal
2. Select Set trigger expressions from the popup menu
3. In the first (left) field, enter `s_THREE` and click OK



Click the Run icon. When signal current_state reaches the value `s_THREE`, the IICE triggers.

Note: Because randomly generated data is applied, the trigger watchpoint (`s_THREE`) 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.



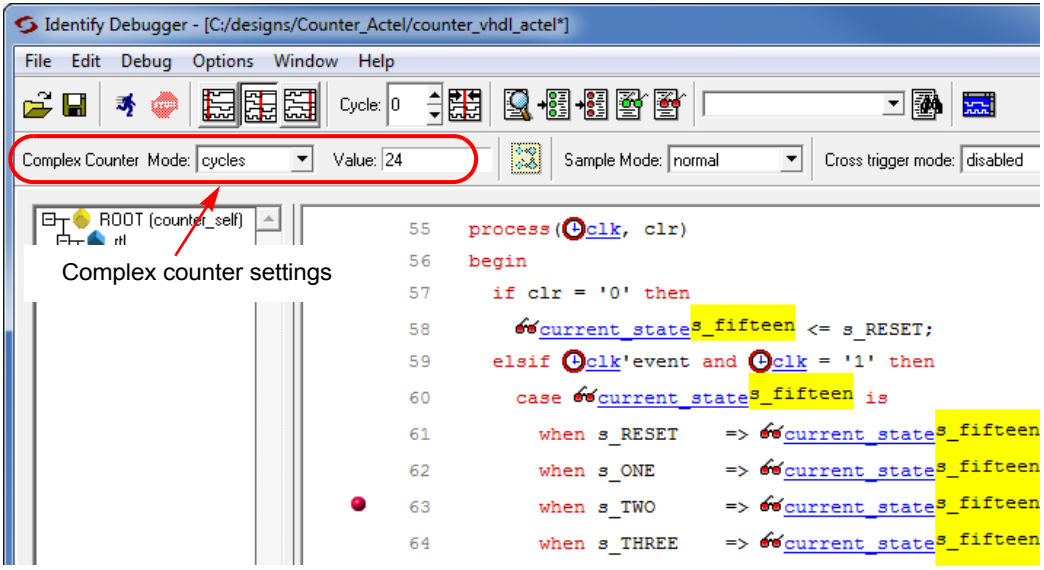
Using the Complex Counter

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 the counter mode to cycles and the counter value to a value greater than 1 (note that you must have previously enabled Complex counter triggering on the IICE Controller tab in the Identify instrumentor).
2. Change the watchpoint of signal current_state to `s_TWO`.
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 has occurred as shown in the following figure.

Note: Because randomly generated data is applied to all instrumented nodes, the results displayed do 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.