

Identify[®] Actel Edition

Quick Start Guide

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



Contents

Quick Start Guide

Before You Start	7
Start the Tool	7
Design Flow with the Identify Tool Set	9
Create an Identify Project	11
Select Desired Instrumentation	13
Configure the IICE	15
Create the Instrumented Design	20
Synthesize and Place and Route the Design	21
Open Project in Identify Debugger	22
Set Trigger Condition	23
Run Debug Hardware	24
View Design Data	25
Communication Errors	26

Quick Start Guide

Before You Start

Before you can start to use the Identify[®] instrumentor (and the Identify[®] debugger), the Identify Actel Edition software must be installed and you must have a license to run the software. The Identify software uses a floating license based on the FLEXnet licensing technology and Synopsys Common Licensing. If you don't have a license, you will be prompted to supply one when you attempt to start the Identify instrumentor.

Start the Tool

The Identify instrumentor (and the Identify debugger) can be started in the graphical mode, shell mode, or, when run in conjunction with the Synplify Pro synthesis tool, can be launched directly from the Synplify Pro user interface.

Graphical Mode

To start the Identify instrumentor or Identify debugger in the stand-alone, graphical mode, select Programs->Synplicity->Identify Instrumentor or Programs->Synplicity->Identify Debugger from the start menu or enter the appropriate command at the command prompt:

```
path_to_identify_install_directory/bin/identify_instrumentor
```

```
path_to_identify_install_directory/bin/identify_debugger
```

The Identify instrumentor or Identify debugger can also be run with a Tcl startup file. To start the tool with a Tcl script, enter the appropriate command:

```
path_to_identify_install_directory/bin/identify_instrumentor -f fileName.tcl  
path_to_identify_install_directory/bin/identify_debugger -f fileName.tcl
```

Shell Mode

Both the Identify instrumentor and Identify debugger can be started in a shell mode and controlled by Tcl commands. To start either tool in the shell mode, enter the appropriate command at the command prompt:

```
path_to_identify_install_directory/bin/identify_instrumentor_shell  
path_to_identify_install_directory/bin/identify_debugger_shell
```

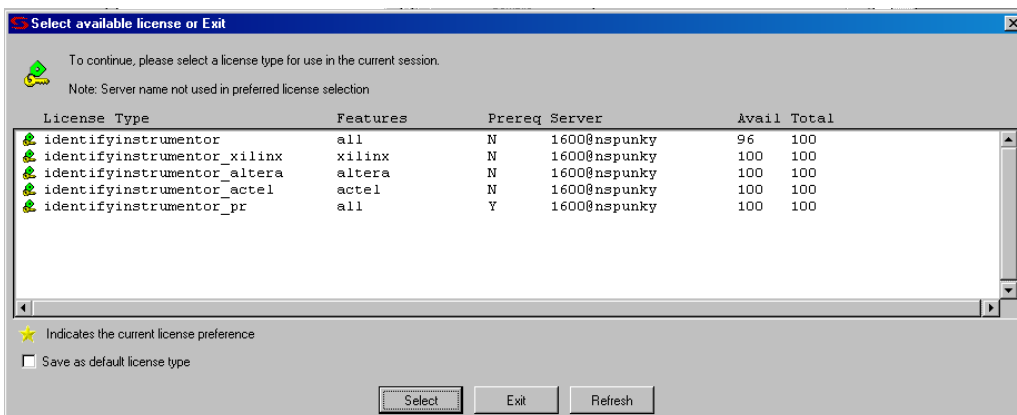
Synplify Pro Synthesis Tool Launch



The Identify instrumentor can be launched in the graphical mode directly from the Synplify Pro graphical user interface by selecting Run->Launch Identify Instrumentor. This method is the simplest and most-direct method of starting the Identify instrumentor. For a full description, see [Launching the Identify Instrumentor from Synplify Pro, on page 11](#).

Vendor Licenses

When you initially start the Identify instrumentor or Identify debugger, you are prompted to select the license type from the licenses available for your configuration. The following figure shows the Select available license dialog box with a full complement of license types.



To select a license type, highlight (click on) the entry and then click the Select button. To avoid being prompted for the license type each time you start the Identify instrumentor or Identify debugger, check the Save as default license type box before clicking the Select button.

After opening the Identify instrumentor or Identify debugger, you can change the license type from within the tool by selecting Help->Preferred License Selection from the menu to display the Preferred license selection dialog box and then selecting a new license type as described in the previous paragraph.

Design Flow with the Identify Tool Set

Because the Identify tool set produces an instrumented version of your design, using these tools should be considered a pre-processing step in your current design flow. This Quick Start guide assumes that you currently have a working FPGA design flow for your Actel device. If not, use a simple design (for example, the supplied tutorial design counter_self) to establish a working flow that effectively moves an HDL design through synthesis, place and route, and programming of the chip. Only after this flow is established should you attempt to debug a design with the Identify tool set.

Before you begin, it is also helpful to note the amount of unused resources available in your original design. The Identify instrumentor provides an estimate of the additional resources necessary for instrumentation in the target device which can help maximize the debugging visibility.

The following figure shows a typical design flow using the Identify software.

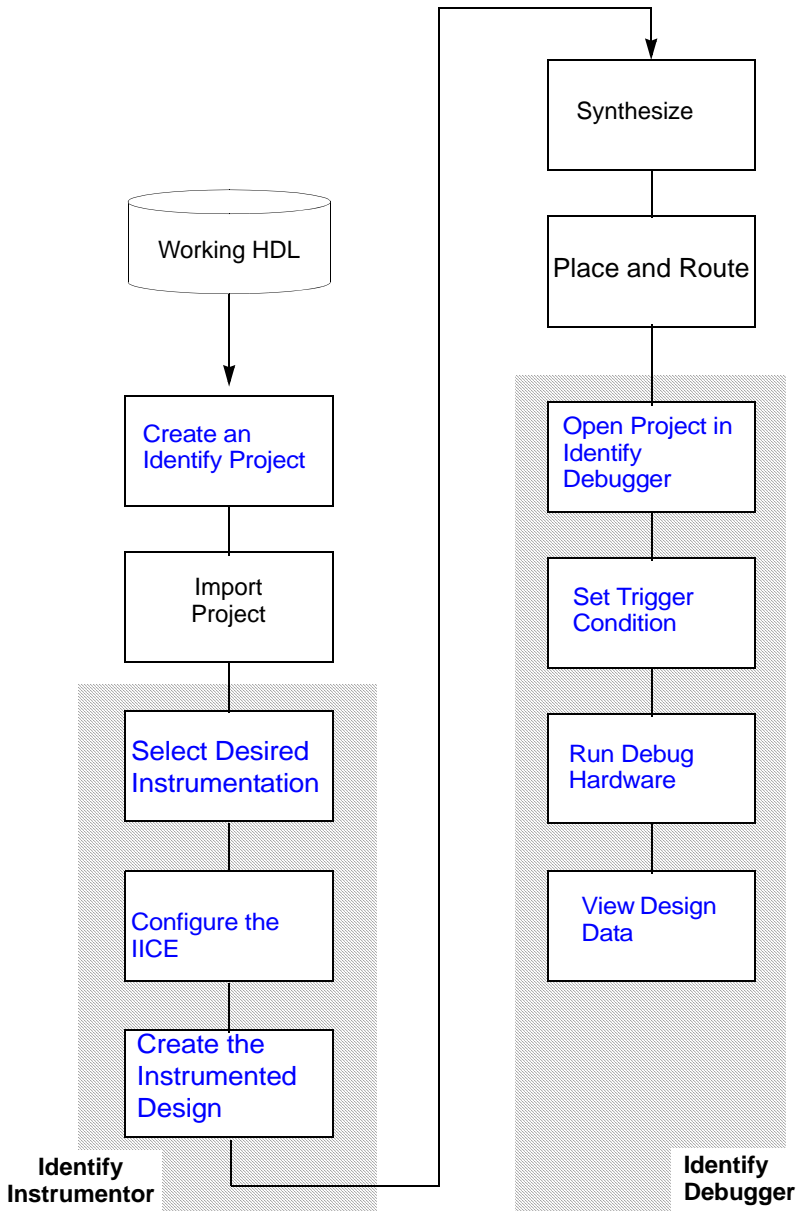


Figure 1: Identify Design Flow

Create an Identify Project

You can create an Identify project by doing any of the following:

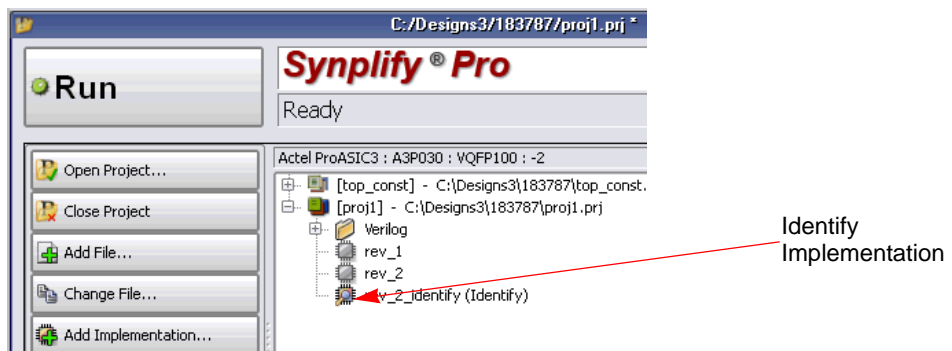
- Open a project in the Synplify Pro synthesis tool and then launch the Identify instrumentor.
- Open the Identify instrumentor and import an existing Synopsys FPGA synthesis project.
- Open the Identify instrumentor in stand-alone mode and add HDL source files.

Launching the Identify Instrumentor from Synplify Pro



For Synplify Pro users, the simplest and most-direct method of creating an Identify project is to open the project in the Synplify Pro project view and:

1. Right click on the implementation to be instrumented and select Project->New Identify Implementation.
2. Verify or set the technology and device mapping options as required. Make sure that a Top Level Module is specified on the Verilog tab or that a Top Level Entity is specified on the VHDL tab (see the implementation option descriptions in the *Synopsys FPGA Synthesis User Guide* for more information). Click OK to close the dialog box; a new Identify implementation is added to the project view.



3. Right click on the Identify implementation and select Launch Identify Instrumentor from the popup menu (you can also click the Launch Identify Instrumentor toolbar icon).

4. If the Launch Identify dialog box is displayed, make sure that the Locate Identify Installation field points to the Identify instrumentor executable (use the browse button to the right of the field if necessary) and make sure that the proper license option is selected. Click OK; if prompted to save your changes, click Save.
5. Pick a license type from the dialog box and click Select.

Clicking the Select button:

- Brings up the Identify instrumentor graphical interface.
- Automatically imports the corresponding project file (*.prj) into the Identify instrumentor tool and opens the project.
- Automatically compiles the project which allows the Identify instrumentor to determine all of the potential locations for instrumenting breakpoints and watchpoints.

Creating an Identify Project from a Synthesis Project

For Synopsys FPGA synthesis tool users, an existing Synopsys FPGA project file (*.prj) can be imported directly into the Identify instrumentor to automatically create a new Identify project. To use this method:

1. Open the Identify instrumentor in stand-alone mode.
2. Pick a license type from the available licenses displayed and click Select.
3. Select File->Import Synplify project to display the Import Synplify/Pro project file dialog box.
4. Navigate to the location of the project (.prj) file and either double click on the project file name or highlight the file and click the Open button.

Opening the project:

- Displays the instrumentation window with the hierarchy browser on the left and the HDL source code on the right.
- Creates an Identify project (.bsp) file in the same project directory.
- Automatically compiles the project which allows the Identify instrumentor to determine all of the potential locations for instrumenting breakpoints and watchpoints.

Creating a New Project from HDL Source

To create a new project from HDL source using the Identify instrumentor graphical user interface (GUI):

1. Open the Identify instrumentor in the GUI.
2. Click the Add Files button to bring up the Add Design Files dialog box.
3. Navigate to the directory containing your HDL design files. All HDL files are listed in the dialog box.
4. Select (highlight) the design file or files to be added to the project and click Open to add the files to the project. Use the Control and Shift keys to add multiple files. The order shown in the list of files is the order in which the files are compiled. Use drag and drop to correct any file-order dependencies.
5. In the Compile Options section of the project window, enter the name of the top-level module or entity into the Top level unit field.
6. Click the Compile button, select Actions->Compile from the menu, or click the Compile current project icon to compile your project.

Note: Alternatively, you can create a TCL script to add the files to your design. For information on scripting and the compile add command, see the reference manual.

Select Desired Instrumentation

When your design compiles successfully, your design with its possible instrumentation is displayed in the instrumentation window. The hierarchy browser on the left shows the design hierarchy and is used to browse your design. Double clicking any hierarchy symbol takes you to its corresponding code location in the HDL source code display on the right.

In the source code display, each potential *watchpoint* is indicated by a glasses icon prefixing the signal name, and each potential *breakpoint* is indicated by a circle in the left margin of the source code.

```
56  begin
57  if  $\delta$ clr = '0' then
58   $\delta$ current_state <= s_RESET;
59  elsif  $\delta$ clk'event and  $\delta$ clk = '1' then
60  case  $\delta$ current_state is
61  when s_RESET    =>  $\delta$ current_state <= s_ONE;
62  when s_ONE      =>  $\delta$ current_state <= s_TWO;
```

Watchpoint →

Breakpoint →

Watchpoints



A watchpoint is instrumented by clicking the watchpoint icon adjacent to the signal name and selecting Sample Only, Trigger Only, or Sample and Trigger from the pop-up menu.

- Sample-only signals are sampled by the debug logic. You can view the data from a sample-only signal, but you cannot use the signal to trigger the hardware during runtime. The lenses of the watchpoint icon are blue for sample-only signals.
- Trigger-only signals are not sampled by the debug logic and have no visibility. These signals are used to create trigger conditions to trigger the debug logic (IICE) during runtime. The lenses of the watchpoint icon are pink for trigger-only signals.
- Sample and trigger signals connect to both the sample logic and trigger logic of the IICE. These signals are visible and can be used to trigger the debug logic. The lenses of the watchpoint icon are green for sample and trigger signals.

Note: Other selections are available for defining watchpoint types on partial buses and on individual fields within a record or a structure; see the user guide for more information.

Breakpoints

Breakpoints are instrumented by clicking on the icon to the left of the breakpoint line. A breakpoint is essentially a control-flow trigger. It is used at runtime to trigger the debug logic based on the flow through case and if/then/else statements. During debug, a breakpoint triggers the IICE whenever the corresponding branch in the code becomes active.

Resource Estimates

As you select each watchpoint and breakpoint, the resource usage is updated in the console window. The usage reported is an estimate of the resources on the target device required to incorporate the additional instrumentation logic. The estimate depends on the number of signals and breakpoints selected for the instrumentation, as well as other IICE settings such as device family and sample depth. Please see the next section, Configure the IICE, for information about these settings.

Note: The Identify instrumentor is not aware of the size of the target device and it is possible to add more instrumentation to the design than will fit on the device. Exceeding the device capacity causes errors during synthesis or place and route.

Configure the IICE

Configuring an IICE to match your debugging requirements involves the setting of a series of IICE parameters. The common IICE parameters are set in the project window and apply to all IICE units defined for the currently-active implementation; the IICE parameters unique to each IICE definition in a multi-IICE configuration are interactively set on one of two IICE Configuration dialog box tabs.

Common IICE Parameters

The common IICE parameters for the currently-active instrumentation are set in the project window after a design is successfully compiled. All IICE units in a multi-IICE configuration share these same parameter values. To redisplay the project window, click the project window tab at the bottom of the window.

- **Device Family** – select the appropriate device technology for your Actel-based design. Notice that when you change the device family, the report window in the IICE editor updates the area estimate for that device family and that the resources are reported in technology-specific terms.

- JTAG Port – make sure the JTAG port selection box is set to builtin. This setting configures the IICE to use the built-in JTAG resources of the device. If access to the built-in JTAG port is not available, you must use the soft JTAG scheme which inserts a JTAG controller into the design and connects it to four user-defined pins (see the user guide for more information about the soft JTAG).

Individual IICE Parameters



The individual parameters for each IICE are defined on the two tabs of the Configure IICE dialog box. To display this dialog box, select Actions->Configure IICE from the menu or click on the Edit IICE settings icon. When setting parameters for an individual IICE in a multi-IICE configuration, use the Current IICE field to specify the target IICE.

IICE Sampler Tab

The IICE Sampler tab controls the size of the IICE sample buffer and defines the sample clock.

IICE Sampler

Current IICE: IICE

IICE Sampler

Buffer type: behavioral

Sample depth: 128

Allow qualified sampling

Allow always-armed triggering

Sample Clock

Sample clock: /clk

Clock edge: Positive Negative

When setting these parameters:

- Buffer type – sets the buffer type; the only supported type is behavioral.
- Sample depth – specifies the desired depth of the sample buffer. This setting controls the size of the sample trace for each signal. Keep in mind that with Identify, you will be able to set very specific triggers and thus may be able to use a smaller trace than required by a logic analyzer. Changing this field also updates the IICE estimate at the bottom of the IICE editor dialog box. For information on using the advanced sampling modes, see the User Guide.

- Allow qualified sampling – when checked, causes the Identify instrumentor to build an IICE block that is capable of performing qualified sampling. When qualified sampling is enabled, one data value is sampled each time the trigger condition is true. With qualified sampling, you can follow the operation of the design over a longer period of time (for example, you can observe the addresses in a number of bus cycles by sampling only one value for each bus cycle instead of a full trace). Using qualified sampling includes a slight area and clock-speed penalty.
- Allow always-armed triggering – when checked, saves the sample buffer for the most recent trigger and waits for the next trigger or until interrupted. When always-armed sampling is enabled, a snapshot is taken each time the trigger condition becomes true. With always-armed triggering, you always acquire the data associated with the last trigger condition prior to the interrupt. This mode is helpful when analyzing a design that uses a repeated pattern as a trigger (for example, bus cycles) and then randomly freezes. You can retrieve the data corresponding to the last time the repeated pattern occurred prior to freezing. Using always-armed sampling includes a slight area and clock-speed penalty.
- Sample clock – determines when signal data is captured by the IICE. The sample clock can be any signal in the design that is a single-bit scalar type. Enter the complete hierarchical path of the signal as the parameter value.

Note: You can also specify the sample clock signal by right-clicking on the watchpoint icon (or signal name) and selecting Sample Clock from the popup menu.

Care must be taken when selecting a sample clock because signals are sampled on an edge of the clock. For the sample values to be valid, the signals being sampled must be stable when the specified edge of the sample clock occurs. Usually, the sample clock is either the same clock that the sampled signals are synchronous with or a multiple of that clock. The sample clock must use a global clock resource of the chip.

Note: If you need help determining the hierarchical path of your clock, try finding it in the HDL source viewer. You may then add and remove it for instrumentation. The full hierarchical path of the signal will be echoed to the TCL command line. Remember that a signal cannot be used as the sample clock if it is instrumented.

- The Clock edge radio buttons determine if samples are taken on the rising (positive) or falling (negative) edge of the sample clock. The default is the positive edge.

IICE Controller Tab

The IICE Controller tab customizes the trigger logic available for triggering the sample buffer.

The screenshot shows the IICE Controller configuration window. At the top, there is a tab labeled "IICE Controller". Below the tab, there is a dropdown menu for "Current IICE:" set to "IICE". The main configuration area is divided into two sections: "IICE Controller" and "IICE Options".

IICE Controller

- Simple triggering
- Complex counter triggering
 - Width: 16
- State Machine triggering
 - Trigger states: 4
 - Trigger conditions: 4
 - Counter width: 16

IICE Options

- Import external trigger signals: 0
- Export IICE trigger signal
- Allow cross-triggering in IICE

Select the Complex counter option and set its size to 32 to instrument a 32-bit counter for use with the triggering logic. This setting will enable you to use advanced trigger operations such as counting the number of trigger events or delaying a trigger event by a set number of clock cycles.

The state machine setting can be used to create fully flexible trigger conditions including capturing samples based on sequences of trigger events. See the user guide for more information on state machine triggering.

The Import external trigger signals option allows triggers from one or more external sources to be imported and configured as a trigger condition for the active IICE. The external source can be a second IICE located on a different device or external logic on the board rather than the result of an Identify instrumentation. Selecting this option automatically selects state-machine triggering.

The Export IICE trigger signal option brings out the IICE's global trigger signal to the top level of your design. This signal can then be used to trigger a logic analyzer or to trigger another IICE.

The Allow cross triggering in IICE option, when enabled, allows the current IICE unit to accept a cross-trigger from another IICE unit.

Create the Instrumented Design



When you are satisfied with the instrumentation, generate the new instrumented design by either selecting File->Save Project from the Identify instrumentor menu bar or clicking on the Save and instrument current project icon on the toolbar. When you generate the new instrumented design:

- the Identify instrumentor writes out your identify project file (*project-Name.bsp*) to the specified directory. This project contains the information about the design and the instrumentation that you have applied to that design; the project can be opened and saved by both the Identify instrumentor and the Identify debugger.
- the Identify instrumentor creates the instrumentation directory named *projectName_instr*. This directory is created in the same directory as the project file and contains the design database used by the Identify debugger and the *instr_sources* subdirectory that holds the newly instrumented sources created by the Identify instrumentor.

Once you have instrumented your design, you will begin the process of implementing the design and then debugging it. Note that any changes to the Identify project, the design files, or the instrumentation can cause the current project to become invalid. Take care not to change the instrumentation or otherwise overwrite the current project. The Identify debugger takes many precautions to ensure correct sample data, and does not allow debugging of a design that has been changed or the viewing of files that have been modified. These types of changes may require you to re-run both synthesis and place and route.

The Identify software allows you to create multiple instrumentations for a single design using another target device as well as using different IICE configuration parameters. Also, you can make incremental changes to the instrumentation set during the debug phase and then only run a partial place and route flow which can drastically reduce the debug cycle time. For information on multiple instrumentations and the incremental flow, see the user guide.

Synthesize and Place and Route the Design

Repeat Original Design Flow with Instrumented Design

When you have successfully created an instrumented design, the design is run through the Synopsys FPGA synthesis design flow. If you are using scripts to run synthesis, it may be necessary to modify the existing scripts to use the new instrumented sources created in the `instr_sources` subdirectory.

Note: Always synthesize and place and route your instrumented design using all of the original constraints and settings.

Read the Project into the Synopsys FPGA Synthesis Tool

When you launch the Identify instrumentor from a Synopsys FPGA synthesis tool, the project file is automatically updated to describe the resulting Identify directory/file structure for the instrumented design. When you synthesize the design, the additional instrumentation logic is automatically included in the design as displayed in the RTL and technology views.

When you run the Identify instrumentor in stand-alone mode (i.e., when the tool is not launched from a Synopsys FPGA synthesis tool), the Identify instrumentor creates a Tcl script file named `synplify.tcl` in the instrumentation directory `projectName_instr`. This file is then imported into the Synopsys FPGA synthesis tool (Run->Run Tcl Script) to create the synthesis project file and load the instrumented design into the synthesis tool.

Add JTAG Clock Constraints

During synthesis, it is important that the JTAG clocks added by Identify are properly constrained (the JTAG clock runs at a very slow speed and must not be optimized to the speed of the design clocks). These clock constraints are handled automatically by the Identify instrumentor by the `syn_dics.sdc` constraint file from the `projectName_instr/instr_sources` directory referenced in the project file.

Any signal with a name that includes `identify_clk` must be constrained to run at 25MHz (40ns) or less. For the place and route tools, the same precautions must be taken to constrain the JTAG clocks to run at 25MHz (40ns) or less.

Program the Instrumented Bit File to the Target Device

When all of the required project files are in place, program the instrumented bit file into the targeted device to load the design logic and the required instrumented logic.

Open Project in Identify Debugger

Any project (*.bsp) that has been instrumented by the Identify instrumentor can be opened in the Identify debugger. The Identify debugger appears similar to the Identify instrumentor except that only instrumented signals and breakpoints are displayed in the Identify debugger. If the Identify debugger is being run on a machine that is different from the host where the design was instrumented, see *Debugging on a Different Machine* in Chapter 7 of the user guide.

Set Trigger Condition

Setting the trigger condition involves setting breakpoints and/or watchpoints in the source code window to trigger the IICE when the associated condition occurs.

Setting Breakpoints

Potential breakpoints are indicated by a green circle in the margin to the left of the source code. Clicking on a breakpoint activates the breakpoint and changes the color of the icon from green to red.

Setting Watchpoints

Watchpoint triggers can be specified on any sampled signal. The watchpoint condition, which is any legal VHDL or Verilog expression that evaluates to a constant, is set through the user interface.

To set a simple watchpoint:

1. Click on the signal
2. Select Set trigger expressions from the popup menu to display the Watchpoint Setup dialog box
3. In the First value field, enter a value for the watch expression
4. Click OK

The setting of the watchpoint trigger is noted by the breakpoint icon next to the watched signal changing from green to red.

Multiple Breakpoints and Watchpoints

When an instrumented design has more than one activated breakpoint, the breakpoint events are ORed together which effectively allows the breakpoints to operate independently – only one activated breakpoint must trigger to cause the sampling buffer to acquire its sample.

When an instrumented design has more than one activated watchpoint, the watchpoint events are ANDed together which effectively causes the watchpoints to be dependent on each other – all activated watchpoint events must occur coincidentally to cause the sampling buffer to acquire its sample.

When an instrumented design has one or more activated breakpoints and one or more activated watchpoints, the result of the OR of the breakpoint events and the result of the AND of the watchpoint events are ANDed together. The result of this AND operation is called the Master Trigger Signal. This ANDing effectively causes the breakpoints and watchpoints to be dependent on each other – one activated breakpoint and all activated watchpoint events must occur coincidentally to cause the sampling buffer to acquire its sample.

Run Debug Hardware

This guide assumes that the Identify debugger communicates with the instrumentation through the same cable that was originally used to program the device. If another communications methodology is being used, see the *Connecting to the Target System* chapter in the user guide. Before running the debug hardware, test your communications setup with the com check command. This command:

- checks the cable connection
- auto-detects the devices on the JTAG chain
- auto-detects the device with instrumentation that matches the current project

If errors are reported, see [Communication Errors, on page 26](#) for possible explanations.



After all of the desired breakpoints and/or watchpoints have been activated, the IICE trigger circuits on the FPGA device are then armed and wait for the watchpointed condition to occur. To arm the IICE trigger circuits on the active IICE, click the Arm current IICE for triggering icon. To arm more than one IICE in a multi-IICE configuration, open the project window in the Identify debugger, check the individual IICE units to be armed, and then click the Run button. Either of these actions downloads the trigger information to the IICE. The IICE now waits for the trigger condition (watchpoint or breakpoint) to occur.

When a watchpoint or breakpoint trigger occurs, sampling is stopped (the hardware continues to run), and the sampled data is transferred back to the debugger where it is displayed in yellow adjacent to the sampled signals in the source code. A small arrow is displayed to the left of the breakpoint or watchpoint icon to indicate the condition that was responsible for the trigger (identifying the trigger condition is important when multiple breakpoints or watchpoints are active).

View Design Data

The sample buffer display can be varied by time. The Cycle display in the middle of the menu bar shows the value zero. This is the point in the sample data buffer where the trigger occurred. By clicking on the up-down arrows on the right, you can increase or decrease the cycle count to show sample buffer 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 buttons and then clicking on the Run button again. The trigger location changes the next time that the IICE triggers.



Early



Middle



Late

Waveform Display

In addition to displaying the sampled data for the selected signals, the Identify debugger can export the sample buffer contents for display in a variety of waveform viewers (GTKWave, Aldec Active-HDL, Novas Debussy).

Select GTKWave Preference

Select Options->Debugger preferences from the menu bar. Verify that GTKWave is the selected choice in the Waveform Viewer Preferences dialog box.

Note: GTKWave is the freeware waveform viewer that is distributed with the Identify tool set.

Click “Waveform” Button



Select Window->Waveform or click the icon labeled Open Waveform Display in the Identify debugger toolbar. A GTKWave waveform display is shown which displays all of the sampled data for each of the sampled signals. The Identify debugger adds two signals to this waveform:

- identify_cycle – an integer that shows the position in the sample buffer. A value of 0 indicates the cycle in which the trigger event occurred.
- identify_sampleclock – a single-bit signal that shows the edges of the reference clock.

Communication Errors

The following are common errors that you may encounter when setting up communications with the Identify debugger.

" ERROR: Communication is stuck at one/zero. Please check the cable connection.

The Identify debugger is unable to communicate with the instrumented device. This error is usually attributed to a cable connection problem. Make sure that the cable is correctly connected between the parallel/USB port and the JTAG port of the board and verify that the cable type is set correctly in the project editor (select File->Edit Project or use the com cabletype Tcl command).

Note: IMPORTANT – This error is often caused by an incorrect parallel port setting. Please try all choices for the communications port setting using either the command line or the selection box in the Identify debugger project editor.

" ERROR: Cannot find valid instrumented design.

This error indicates that the design on the programmable chip is NOT the instrumented version of the design. Verify that the bit file you are programming is actually created from the instrumented sources and that the debug logic (IICE) has not been removed during implementation. Verification can often be done by searching the intermediate place and route files for the word “identify.”

" ERROR: Instrumented design on FPGA differs from design loaded into Identify Debugger.

This error indicates that the Identify debugger cannot find a device in the JTAG chain that has been instrumented by the Identify instrumentor. In this case, the ID of the instrumentation does not match the ID of the currently loaded project. Please verify that the correct project is loaded for the corresponding bit file. The error occurs when the project is re-instrumented without regenerating a bit file. If you have changed the design or its instrumentation, you must create a new bit file before debugging the design.

" ERROR: No hardware devices were found. Please check the cable connection.

This error indicates that no devices are visible in the JTAG identification register chain. The error is usually caused by a bad cable connection or an incorrect cable type setting.

" ERROR: The hardware has not seen an active clock edge of the sample clock.

This error usually indicates that the Identify hardware is functioning correctly and that the Identify debugger is able to communicate with the device, but that the sample clock is not being toggled. This error can be caused if the wrong clock is chosen in the Identify instrumentor. Please verify that the correct sample clock is selected using the `iice clock` command. Also check that the clock signal is using the global clock resources of the chip.

Note: This error often occurs when the clock signal is not assigned correctly to the clock pin on the board. Verify that, during placement, the clock signal (sample clock) is correctly assigned to the chip pin that is connected to the clock oscillator on the board.

" ERROR: Hardware driver failure.

This error usually indicates that the correct port driver is not installed on the debug system. Please see the release notes for help installing the port driver.

Clock Skew

When the data returned from the Identify debugger appears incorrect or does not properly relate to the given trigger condition, there may be an issue with clock skew on the JTAG clock. Make sure that the `identify_clk` signal is using the global clock resources on the chip.

Debug Mode

If you are experiencing a communication error that is not listed above or if you have not been able to resolve the error, contact your customer support group. It may be helpful to run the Identify debugger in “debug” mode as outlined below:

1. If open, close the debugger.
2. Right-click the Identify debugger shortcut on the desktop and select Properties from the popup menu to display the Properties dialog box.
3. Append the `-debug` flag to the path to the executable in the target field.
4. Restart the Identify debugger and reopen the project.
5. Enter the following two commands at the command prompt in the Console window:

```
chain clear  
com check
```
6. Make a copy of the log file and send it with any other important details about your particular setup/flow as well as the Identify project file (*.bsp) to your customer support group.