QII53021-12.0.0

This chapter provides detailed instructions about how to use the In-System Sources and Probes Editor and Tcl scripting in the Quartus® II software to debug your design.

Traditional debugging techniques often involve using an external pattern generator to exercise the logic and a logic analyzer to study the output waveforms during run time. The SignalTap® II Logic Analyzer and SignalProbe allow you to read or "tap" internal logic signals during run time as a way to debug your logic design. You can make the debugging cycle more efficient when you can drive any internal signal manually within your design, which allows you to perform the following actions:

■ Force the occurrence of trigger conditions set up in the SignalTap II Logic Analyzer

■ Create simple test vectors to exercise your design without using external test equipment

■ Dynamically control run time control signals with the JTAG chain

The In-System Sources and Probes Editor in the Quartus II software extends the portfolio of verification tools, and allows you to easily control any internal signal and provides you with a completely dynamic debugging environment. Coupled with either the SignalTap II Logic Analyzer or SignalProbe, the In-System Sources and Probes Editor gives you a powerful debugging environment in which to generate stimuli and solicit responses from your logic design.

The Virtual JTAG Megafunction and the In-System Memory Content Editor also give you the capability to drive virtual inputs into your design. The Quartus II software offers a variety of on-chip debugging tools. For an overview and comparison of all the tools available in the Quartus II software on-chip debugging tool suite, refer to *Section IV. System Debugging Tools* in volume 3 of the *Quartus II Handbook*.

Overview

This chapter includes the following topics:

The In-System Sources and Probes Editor consists of the ALTSOURCE_PROBE megafunction and an interface to control the ALTSOURCE_PROBE megafunction instances during run time. Each ALTSOURCE_PROBE megafunction instance provides you with source output ports and probe input ports, where source ports drive selected signals and probe ports sample selected signals. When you compile

Twitter   Feedback Subscribe

your design, the ALTSOURCE_PROBE megafunction sets up a register chain to either drive or sample the selected nodes in your logic design. During run time, the In-System Sources and Probes Editor uses a JTAG connection to shift data to and from the ALTSOURCE_PROBE megafunction instances. Figure 16–1 shows a block diagram of the components that make up the In-System Sources and Probes Editor.

Figure 16–1. In-System Sources and Probes Editor Block Diagram
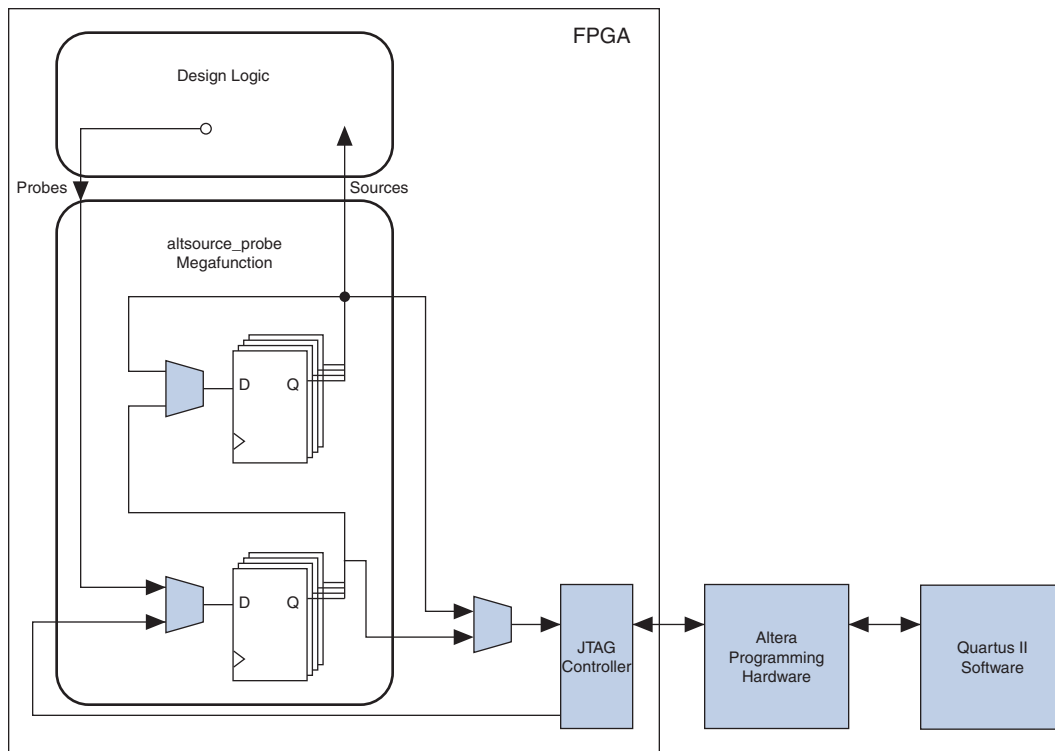
The ALTSOURCE_PROBE megafunction hides the detailed transactions between the JTAG controller and the registers instrumented in your design to give you a basic building block for stimulating and probing your design. Additionally, the In-System Sources and Probes Editor provides single-cycle samples and single-cycle writes to selected logic nodes. You can use this feature to input simple virtual stimuli and to capture the current value on instrumented nodes. Because the In-System Sources and Probes Editor gives you access to logic nodes in your design, you can toggle the inputs of low-level components during the debugging process. If used in conjunction with the SignalTap II Logic Analyzer, you can force trigger conditions to help isolate your problem and shorten your debugging process.

The In-System Sources and Probes Editor allows you to easily implement control signals in your design as virtual stimuli. This feature can be especially helpful for prototyping your design, such as in the following operations:

■ Creating virtual push buttons

■ Creating a virtual front panel to interface with your design

■ Emulating external sensor data

■ Monitoring and changing run time constants on the fly

The In-System Sources and Probes Editor supports Tcl commands that interface with all your ALTSOURCE_PROBE megafunction instances to increase the level of automation.

Hardware and Software Requirements

The following components are required to use the In-System Sources and Probes Editor:

■ Quartus II software

or

■ Quartus II Web Edition (with the TalkBack feature turned on)

■ Download Cable (USB-Blaster™ download cable or ByteBlaster™ cable)

■ Altera® development kit or user design board with a JTAG connection to device under test
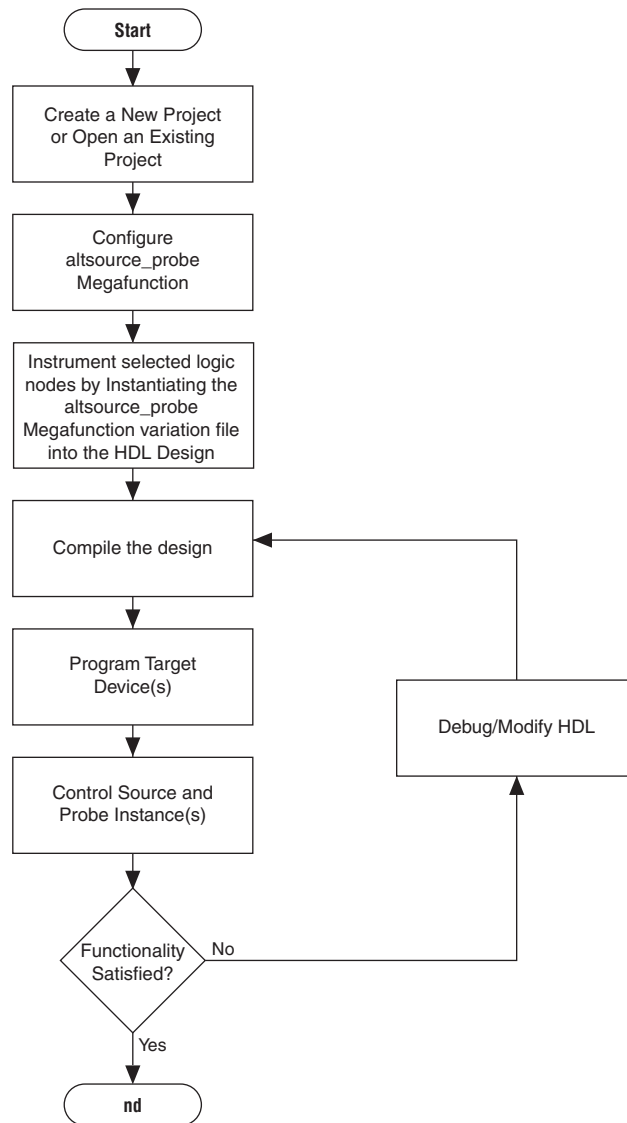
The In-System Sources and Probes Editor supports the following device families:

■ Arria® GX

■ Stratix® series

■ Cyclone® series

■ MAX® series

Design Flow Using the In-System Sources and Probes Editor

The In-System Sources and Probes Editor supports an RTL flow. Signals that you want to view in the In-System Sources and Probes editor are connected to an instance of the ALTSOURCE_PROBE megafunction. After you compile the design, you can control each ALTSOURCE_PROBE instance via the **In-System Sources and Probes Editor** pane or via a Tcl interface. The complete design flow is shown in Figure 16–2.

Figure 16–2. FPGA Design Flow Using the In-System Sources and Probes Editor

Configuring the ALTSOURCE_PROBE Megafunction

To use the In-System Sources and Probes Editor in your design, you must first instantiate the ALTSOURCE_PROBE megafunction variation file. You can configure the ALTSOURCE_PROBE megafunction with the MegaWizard™ Plug-In Manager. Each source or probe port can be up to 256 bits. You can have up to 128 instances of the ALTSOURCE_PROBE megafunction in your design.

To configure the ALTSOURCE_PROBE megafunction, performing the following steps:

1. On the Tools menu, click **MegaWizard Plug-In Manager**.

2. Select **Create a new custom megafunction variation**.

3. Click **Next**.

4. On page 2a of the MegaWizard Plug-In Manager, make the following selections:

   a. In the **Installed Plug-Ins** list, expand the **JTAG-accessible Extensions** folder and select **In-System Sources and Probes**.

   ☞   Verify that the currently selected device family matches the device you are targeting.

   b. Select an output file type and enter the name of the ALTSOURCE_PROBE megafunction. You can choose AHDL (**.tdf**), VHDL (**.vhd**), or Verilog HDL (**.v**) as the output file type.

5. Click **Next**.

6. On page 3 of the MegaWizard Plug-In Manager, make the following selections:

   a. Under **Do you want to specify an Instance Index?**, turn on **Yes**.

   b. Specify the **'Instance ID'** of this instance.

   c. Specify the width of the probe port. The width can be from 0 bit to 256 bits.

   d. Specify the width of the source port. The width can be from 0 bit to 256 bits.

7. On page 3 of the MegaWizard Plug-In Manager, you can click **Advanced Options** and specify other options, including the following:

   ■ **What is the initial value of the source port, in hexadecimal?**—Allows you to specify the initial value driven on the source port at run time.

   ■ **Write data to the source port synchronously to the source clock**—Allows you to synchronize your source port write transactions with the clock domain of your choice.

   ■ **Create an enable signal for the registered source port**—When turned on, creates a clock enable input for the synchronization registers. You can turn on this option only when the **Write data to the source port synchronously to the source clock** option is turned on.

☞   The In-System Sources and Probes Editor does not support simulation. You must remove the ALTSOURCE_PROBE megafunction instantiation before you create a simulation netlist.

Instantiating the ALTSOURCE_PROBE Megafunction

The MegaWizard Plug-In Manager produces the necessary variation file and the instantiation template based on your inputs to the MegaWizard. Use the template to instantiate the ALTSOURCE_PROBE megafunction variation file in your design. The port information is shown in Table 16–1.

Table 16–1. ALTSOURCE_PROBE Megafunction Port Information

Port Name	Required?	Direction	Comments
probe[]	No	Input	The outputs from your design.
source_clk	No	Input	Source Data is written synchronously to this clock. This input is required if you turn on **Source Clock** in the **Advanced Options** box in the MegaWizard Plug-In Manager.
source_ena	No	Input	Clock enable signal for source_clk. This input is required if specified in the **Advanced Options** box in the MegaWizard Plug-In Manager.
source[]	No	Output	Used to drive inputs to user design.

You can include up to 128 instances of the ALTSOURCE_PROBE megafunction in your design, if your device has available resources. Each instance of the ALTSOURCE_PROBE megafunction uses a pair of registers per signal for the width of the widest port in the megafunction. Additionally, there is some fixed overhead logic to accommodate communication between the ALTSOURCE_PROBE instances and the JTAG controller. You can also specify an additional pair of registers per source port for synchronization.

Compiling the Design

When you compile your design with the In-System Sources and Probes megafunction instantiated, an instance of the ALTSOURCE_PROBE and SLD_HUB instances are added to your compilation hierarchy automatically. These instances provide communication between the JTAG controller and your instrumented logic.

You can modify the number of connections to your design by editing the ALTSOURCE_PROBE megafunction. To open the design instance you want to modify in the MegaWizard Plug-In Manager, double-click the instance in the Project Navigator. You can then modify the connections in the HDL source file. You must recompile your design after you make changes.

You can use the Quartus II incremental compilation feature to reduce compilation time. Incremental compilation allows you to organize your design into logical partitions. During recompilation of a design, incremental compilation preserves the compilation results and performance of unchanged partitions and reduces design iteration time by compiling only modified design partitions.

For more information about the Quartus II incremental compilation feature, refer to the *Quartus II Incremental Compilation for Hierarchical and Team-Based Design* chapter in volume 1 of the *Quartus II Handbook*.

Running the In-System Sources and Probes Editor

The In-System Sources and Probes Editor gives you control over all ALTSOURCE_PROBE megafunction instances within your design. The editor allows you to view all available run time controllable instances of the ALTSOURCE_PROBE megafunction in your design, provides a push-button interface to drive all your source nodes, and provides a logging feature to store your probe and source data.

To run the In-System Sources and Probes Editor, on the **Tools** menu, click **In-System Sources and Probes Editor.**

The In-System Sources and Probes Editor contains three panes:

- **JTAG Chain Configuration**—Allows you to specify programming hardware, device, and file settings that the In-System Sources and Probes Editor uses to program and acquire data from a device.

- **Instance Manager**—Displays information about the instances generated when you compile a design, and allows you to control data that the In-System Sources and Probes Editor acquires.

- **In-System Sources and Probes Editor**—Logs all data read from the selected instance and allows you to modify source data that is written to your device.

When you use the In-System Sources and Probes Editor, you do not need to open a Quartus II software project. The In-System Sources and Probes Editor retrieves all instances of the ALTSOURCE_PROBE megafunction by scanning the JTAG chain and sending a query to the device selected in the **JTAG Chain Configuration** pane. You can also use a previously saved configuration to run the In-System Sources and Probes Editor.

Each **In-System Sources and Probes Editor** pane can access the ALTSOURCE_PROBE megafunction instances in a single device. If you have more than one device containing megafunction instances in a JTAG chain, you can launch multiple **In-System Sources and Probes Editor** panes to access the megafunction instances in each device.

Programming Your Device With JTAG Chain Configuration

After you compile your project, you must configure your FPGA before you use the In-System Sources and Probes Editor. To configure a device to use with the In-System Sources and Probes Editor, perform the following steps:

1. Open the In-System Sources and Probes Editor.

2. In the **JTAG Chain Configuration** pane, point to **Hardware,** and then select the hardware communications device. You may be prompted to configure your hardware; in this case, click **Setup**.

3. From the **Device** list, select the FPGA device to which you want to download the design (the device may be automatically detected). You may need to click **Scan Chain** to detect your target device.

4. In the **JTAG Chain Configuration** pane, click to browse for the SRAM Object File (**.sof**) that includes the In-System Sources and Probes instance or instances. (The **.sof** may be automatically detected).

5. Click **Program Device** to program the target device.

Instance Manager

The **Instance Manager** pane provides a list of all ALTSOURCE_PROBE instances in the design and allows you to configure how data is acquired from or written to those instances.

The following buttons and sub-panes are provided in the **Instance Manager** pane:

■ **Read Probe Data**—Samples the probe data in the selected instance and displays the probe data in the **In-System Sources and Probes Editor** pane.

■ **Continuously Read Probe Data**—Continuously samples the probe data of the selected instance and displays the probe data in the **In-System Sources and Probes Editor** pane; you can modify the sample rate via the **Probe read interval** setting.

■ **Stop Continuously Reading Probe Data**—Cancels continuous sampling of the probe of the selected instance.

■ **Write Source Data**—Writes data to all source nodes of the selected instance.

■ **Probe Read Interval**—Displays the sample interval of all the In-System Sources and Probe instances in your design; you can modify the sample interval by clicking **Manual.**

■ **Event Log**—Controls the event log in the **In-System Sources and Probes Editor** pane.

■ **Write Source Data**—Allows you to manually or continuously write data to the system.

The status of each instance is also displayed beside each entry in the **Instance Manager** pane. The status indicates if the instance is **Not running Offloading data**, **Updating data**, or if an **Unexpected JTAG communication error** occurs. This status indicator provides information about the sources and probes instances in your design.

In-System Sources and Probes Editor Pane

The **In-System Sources and Probes Editor** pane allows you to view data from all sources and probes in your design. The data is organized according to the index number of the instance. The editor provides an easy way to manage your signals, and allows you to rename signals or group them into buses. All data collected from in-system source and probe nodes is recorded in the event log and you can view the data as a timing diagram.

Reading Probe Data

You can read data by selecting the ALTSOURCE_PROBE instance in the **Instance Manager** pane and clicking **Read Probe Data**. This action produces a single sample of the probe data and updates the data column of the selected index in the **In-System Sources and Probes Editor** pane. You can save the data to an event log by turning on the **Save data to event log** option in the **Instance Manager** pane.

If you want to sample data from your probe instance continuously, in the **Instance Manager** pane, click the instance you want to read, and then click **Continuously read probe data**. While reading, the status of the active instance shows **Unloading**. You can read continuously from multiple instances.

You can access read data with the shortcut menus in the **Instance Manager** pane.

To adjust the probe read interval, in the **Instance Manager** pane, turn on the **Manual** option in the **Probe read interval** sub-pane, and specify the sample rate in the text field next to the **Manual** option. The maximum sample rate depends on your computer setup. The actual sample rate is shown in the **Current interval** box. You can adjust the event log window buffer size in the **Maximum Size** box.

Writing Data

To modify the source data you want to write into the ALTSOURCE_PROBE instance, click the name field of the signal you want to change. For buses of signals, you can double-click the data field and type the value you want to drive out to the ALTSOURCE_PROBE instance. The In-System Sources and Probes Editor stores the modified source data values in a temporary buffer. Modified values that are not written out to the ALTSOURCE_PROBE instances appear in red. To update the ALTSOURCE_PROBE instance, highlight the instance in the **Instance Manager** pane and click **Write source data**. The **Write source data** function is also available via the shortcut menus in the **Instance Manager** pane.

The In-System Sources and Probes Editor provides the option to continuously update each ALTSOURCE_PROBE instance. Continuous updating allows any modifications you make to the source data buffer to also write immediately to the ALTSOURCE_PROBE instances. To continuously update the ALTSOURCE_PROBE instances, change the **Write source data** field from **Manually** to **Continuously**.

Organizing Data

The **In-System Sources and Probes Editor** pane allows you to group signals into buses, and also allows you to modify the display options of the data buffer.

To create a group of signals, select the node names you want to group, right-click and select **Group**. You can modify the display format in the Bus Display Format and the Bus Bit order shortcut menus.

The **In-System Sources and Probes Editor** pane allows you to rename any signal. To rename a signal, double-click the name of the signal and type the new name.

The event log contains a record of the most recent samples. The buffer size is adjustable up to 128k samples. The time stamp for each sample is logged and is displayed above the event log of the active instance as you move your pointer over the data samples.

You can save the changes that you make and the recorded data to a Sources and Probes File (**.spf**). To save changes, on the File menu, click **Save**. The file contains all the modifications you made to the signal groups, as well as the current data event log.

Tcl interface for the In-System Sources and Probes Editor

To support automation, the In-System Sources and Probes Editor supports the procedures described in this chapter in the form of Tcl commands. The Tcl package for the In-System Sources and Probes Editor is included by default when you run `quartus_stp`.

The Tcl interface for the In-System Sources and Probes Editor provides a powerful platform to help you debug your design. The Tcl interface is especially helpful for debugging designs that require toggling multiple sets of control inputs. You can combine multiple commands with a Tcl script to define a custom command set.

For more information about Tcl scripting, refer to the *Tcl Scripting* chapter in volume 2 of the *Quartus II Handbook*. For more information about settings and constraints in the Quartus II software, refer to the *Quartus II Settings File Manual*. For more information about command-line scripting, refer to the *Command-Line Scripting* chapter in volume 2 of the *Quartus II Handbook*.

Table 16–2 shows the Tcl commands you can use instead of the In-System Sources and Probes Editor.

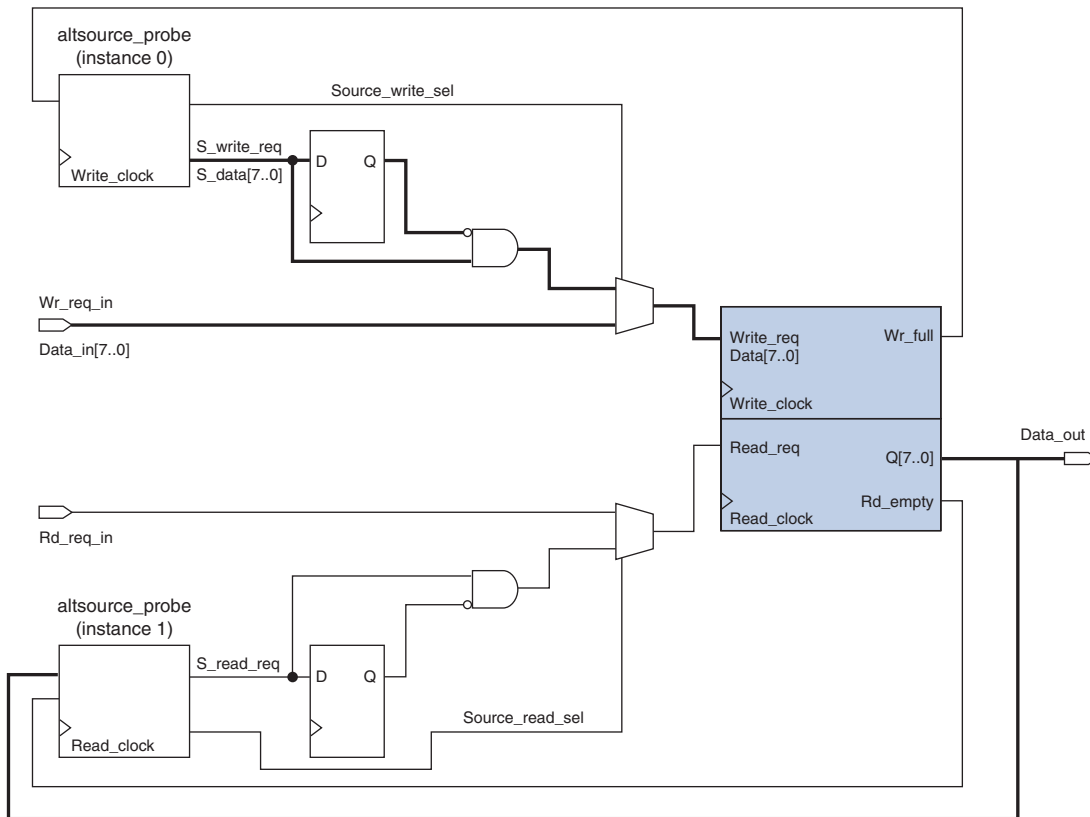**Table 16–2. In-System Sources and Probes Tcl Commands**

Command	Argument	Description
`start_insystem_source_probe`	`-device_name <device name>` `-hardware_name <hardware name>`	Opens a handle to a device with the specified hardware. Call this command before starting any transactions.
`get_insystem_source_probe_instance_info`	`-device_name <device name>` `-hardware_name <hardware name>`	Returns a list of all `ALTSOURCE_PROBE` instances in your design. Each record returned is in the following format: {*<instance Index>*, *<source width>*, *<probe width>*, *<instance name>*}
`read_probe_data`	`-instance_index <instance_index>` `-value_in_hex (optional)`	Retrieves the current value of the probe. A string is returned that specifies the status of each probe, with the MSB as the left-most bit.
`read_source_data`	`-instance_index <instance_index>` `-value_in_hex (optional)`	Retrieves the current value of the sources. A string is returned that specifies the status of each source, with the MSB as the left-most bit.
`write_source_data`	`-instance_index <instance_index>` `-value <value>` `-value_in_hex (optional)`	Sets the value of the sources. A binary string is sent to the source ports, with the MSB as the left-most bit.
`end_interactive_probe`	None	Releases the JTAG chain. Issue this command when all transactions are finished.

Example 16–1 shows an excerpt from a Tcl script with procedures that control the ALTSOURCE_PROBE instances of the design as shown in Figure 16–3. The example design contains a DCFIFO with ALTSOURCE_PROBE instances to read from and write to the DCFIFO. A set of control muxes are added to the design to control the flow of data to the DCFIFO between the input pins and the ALTSOURCE_PROBE instances. A pulse generator is added to the read request and write request control lines to guarantee a single sample read or write. The ALTSOURCE_PROBE instances,

Chapter 16: Design Debugging Using In-System Sources and Probes
Tcl interface for the In-System Sources and Probes Editor

16–11

when used with the script in Example 16–1, provide visibility into the contents of the FIFO by performing single sample write and read operations and reporting the state of the full and empty status flags.

Use the Tcl script in debugging situations to either empty or preload the FIFO in your design. For example, you can use this feature to preload the FIFO to match a trigger condition you have set up within the SignalTap II Logic Analyzer.

Figure 16–3. A DCFIFO Example Design Controlled by the Tcl Script in Example 16–1

16–12

Chapter 16: Design Debugging Using In-System Sources and Probes
Tcl interface for the In-System Sources and Probes Editor

Example 16–1. Tcl Script Procedures for Reading and Writing to the DCFIFO in Figure 16–3 (Part 1 of 2)

```
## Setup USB hardware  - assumes only USB Blaster is installed and
## an FPGA is the only device in the JTAG chain

set usb [lindex [get_hardware_names] 0]
set device_name [lindex [get_device_names -hardware_name $usb] 0]
## write procedure :   argument value is integer

proc write {value} {

    global device_name usb
    variable full

    start_insystem_source_probe -device_name $device_name -hardware_name $usb

    #read full flag
    set full [read_probe_data -instance_index 0]


    if {$full == 1} {end_insystem_source_probe
    return "Write Buffer Full"
    }
```

Example 16–1. Tcl Script Procedures for Reading and Writing to the DCFIFO in Figure 16–3   (Part 2 of 2)

```
    ##toggle select line, drive value onto port, toggle enable
    ##bits 7:0 of instance 0 is S_data[7:0]; bit 8 = S_write_req;
    ##bit 9 = Source_write_sel

    ##int2bits is custom procedure that returns a bitstring from an integer
    ## argument


    write_source_data -instance_index 0 -value /[int2bits [expr 0x200 | $value]]
    write_source_data -instance_index 0 -value [int2bits [expr 0x300 | $value]]

    ##clear transaction

    write_source_data -instance_index 0 -value 0

    end_insystem_source_probe
}

proc read {} {

    global device_name usb
    variable empty
    start_insystem_source_probe -device_name $device_name -hardware_name $usb

    ##read empty flag : probe port[7:0] reads FIFO output; bit 8 reads empty_flag

    set empty [read_probe_data -instance_index 1]


    if {[regexp {1........} $empty]} { end_insystem_source_probe
    return "FIFO empty" }

    ## toggle select line for read transaction
    ## Source_read_sel = bit 0; s_read_reg = bit 1

    ## pulse read enable on DC FIFO
    write_source_data -instance_index 1 -value 0x1 -value_in_hex
    write_source_data -instance_index 1 -value 0x3 -value_in_hex

    set x [read_probe_data -instance_index 1 ]

    end_insystem_source_probe

    return $x
}
```

Design Example: Dynamic PLL Reconfiguration

The In-System Sources and Probes Editor can help you create a virtual front panel during the prototyping phase of your design. You can create relatively simple, high functioning designs of in a short amount of time. The following PLL reconfiguration example demonstrates how to use the In-System Sources and Probes Editor to provide a GUI to dynamically reconfigure a Stratix PLL.

Stratix PLLs allow you to dynamically update PLL coefficients during run time. Each enhanced PLL within the Stratix device contains a register chain that allows you to modify the pre-scale counters (m and n values), output divide counters, and delay counters. In addition, the ALTPLL_RECONFIG megafunction provides an easy interface to access the register chain counters. The ALTPLL_RECONFIG

megafunction provides a cache that contains all modifiable PLL parameters. After you update all the PLL parameters in the cache, the ALTPLL_RECONFIG megafunction drives the PLL register chain to update the PLL with the updated parameters. Figure 16–4 shows a Stratix-enhanced PLL with reconfigurable coefficients.

☞ Stratix II and Stratix III devices also allow you to dynamically reconfigure PLL parameters. For more information about these families, refer to the appropriate data sheet. For more information about dynamic PLL reconfiguration, refer to *AN 282: Implementing PLL Reconfiguration in Stratix & Stratix GX Devices* or *AN 367: Implementing PLL Reconfiguration in Stratix II Devices*.

Figure 16–4. Stratix-Enhanced PLL with Reconfigurable Coefficients
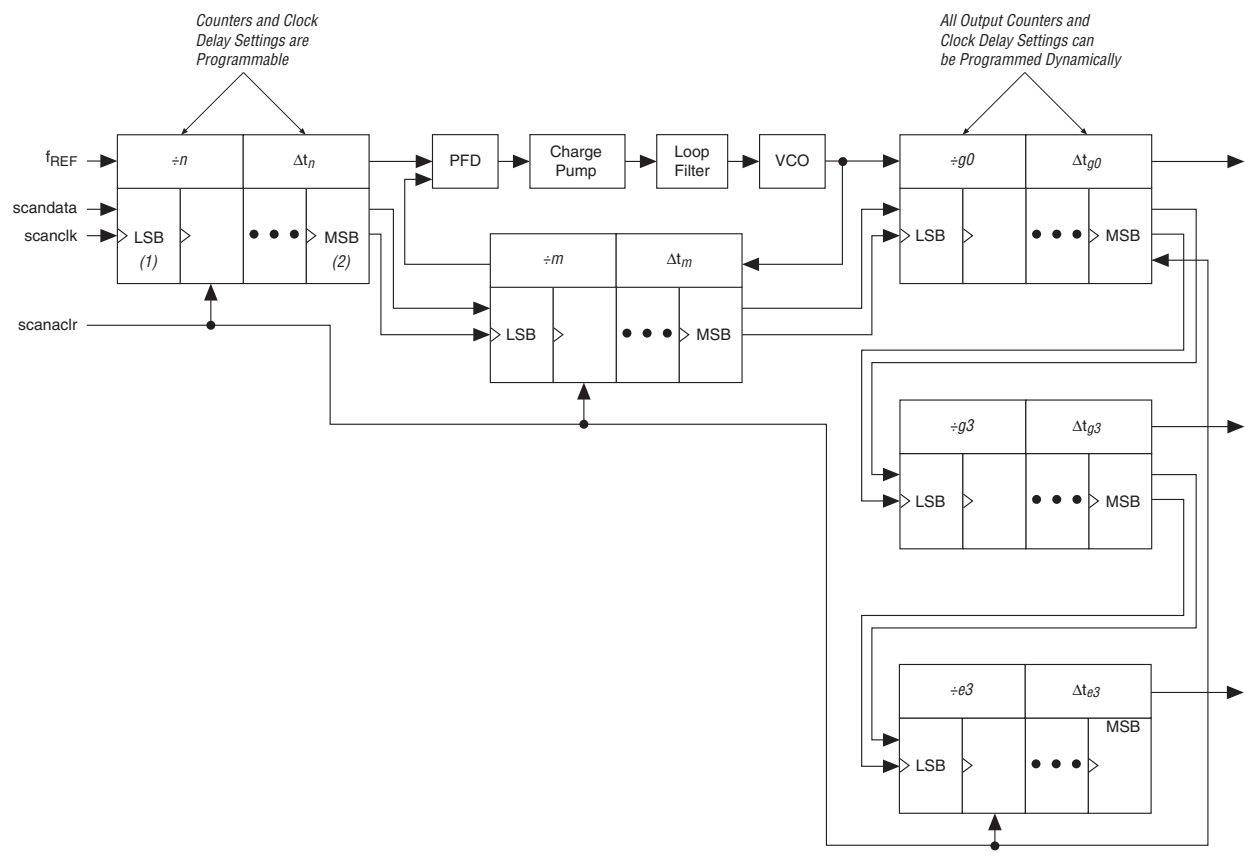
The following design example uses an ALTSOURCE_PROBE instance to update the PLL parameters in the ALTPLL_RECONFIG megafunction cache. The ALTPLL_RECONFIG megafunction connects to an enhanced PLL in a Stratix FPGA to drive the register chain containing the PLL reconfigurable coefficients. This design example uses a Tcl/Tk script to generate a GUI where you can enter in new m and n values for the enhanced PLL. The Tcl script extracts the m and n values from the GUI, shifts the values out to the ALTSOURCE_PROBE instances to update the values in the

ALTPLL_RECONFIG megafunction cache, and asserts the reconfiguration signal on the ALTPLL_RECONFIG megafunction. The reconfiguration signal on the ALTPLL_RECONFIG megafunction starts the register chain transaction to update all PLL reconfigurable coefficients. A block diagram of a design example is shown in Figure 16–5. The Tk GUI is shown in Figure 16–6.

Figure 16–5. Block Diagram of Dynamic PLL Reconfiguration Design Example
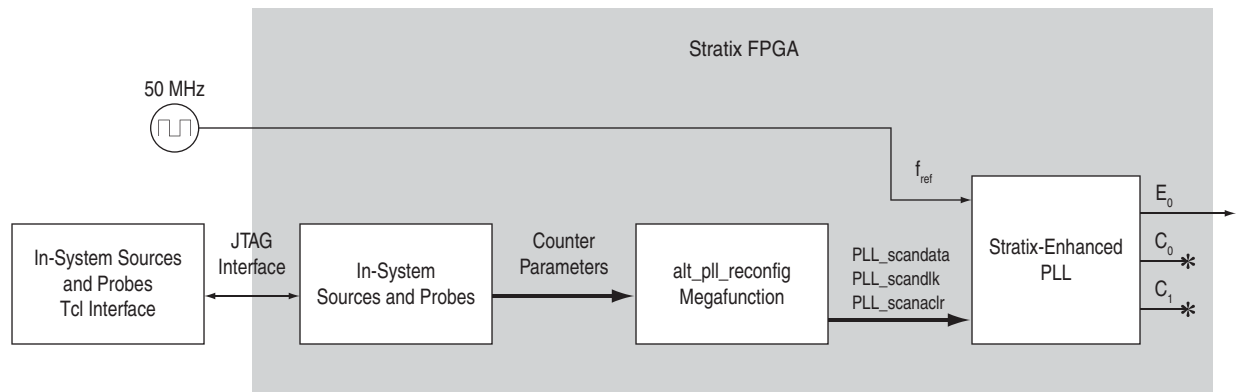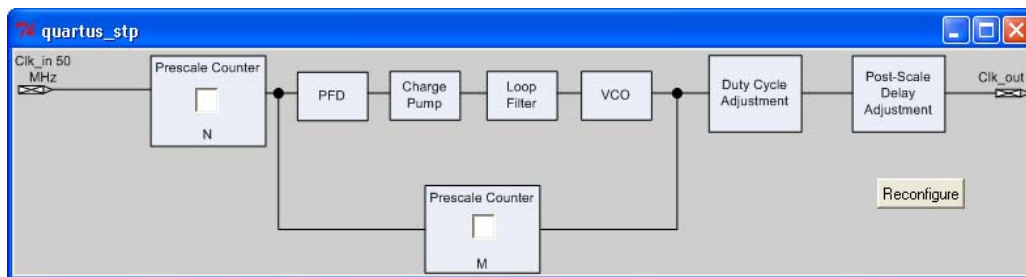

Figure 16–6. Interactive PLL Reconfiguration GUI Created with Tk and In-System Sources and Probes Tcl Package

This design example was created using a Nios® II Development Kit, Stratix Edition. The file **sourceprobe_DE_dynamic_pll.zip** contains all the necessary files for running this design example, including the following:

■ **Readme.txt**—A text file that describes the files contained in the design example and provides instructions about running the Tk GUI shown in Figure 16–6.

■ **Interactive_Reconfig.qar**—The archived Quartus II project for this design example.

Download the **In-System Sources and Probes Example** from the On-chip Debugging Design Examples page of the Altera website.

Conclusion

The In-System Sources and Probes Editor provides stimuli and receives responses from the target design during run time. With the simple and intuitive interface, you can add virtual inputs to your design during run time without using external equipment. When used in conjunction with the SignalTap II Logic Analyzer, you can use the In-System Sources and Probes Editor to obtain greater control of the signals in your design, and thus help shorten the verification cycle.

Document Revision History

Table 16–3 shows the revision history for this chapter.

Table 16–3. Document Revision History

Date	Version	Changes
June 2012	12.0.0	Removed survey link.
November 2011	10.1.1	Template update.
December 2010	10.1.0	Minor corrections. Changed to new document template.
July 2010	10.0.0	Minor corrections.
November 2009	9.1.0	■ Removed references to obsolete devices. ■ Style changes.
March 2009	9.0.0	No change to content.
November 2008	8.1.0	Changed to 8-1/2 x 11 page size. No change to content.
May 2008	8.0.0	■ Documented that this feature does not support simulation on page 17–5 ■ Updated Figure 17–8 for Interactive PLL reconfiguration manager ■ Added hyperlinks to referenced documents throughout the chapter ■ Minor editorial updates

For previous versions of the *Quartus II Handbook*, refer to the Quartus II Handbook Archive.