AN 964: Signal Tap Tutorial for Intel® Agilex™ Partial Reconfiguration Design

Updated for Intel® Quartus® Prime Design Suite: **21.4**

Contents

1. Tutorial Overview

This document demonstrates how to debug an Intel® Agilex™ Partial Reconfiguration design with the Signal Tap Logic Analyzer.

Partial Reconfiguration is an advanced design flow that allows you to reconfigure a portion of the FPGA dynamically, while the remaining FPGA design continues to function. You can define multiple personas to occupy the same design region, without impacting operation in other regions.

This application note extends the Partial Reconfiguration (PR) process described in *AN 953: Partially Reconfiguring a Design: on an Intel Agilex F-Series FPGA Development Board* to the Signal Tap logic analyzer debugging environment.

The Signal Tap logic analyzer, available in the Intel Quartus® Prime software, captures and displays the real-time signal behavior in an Intel FPGA design. Use the Signal Tap logic analyzer to probe and debug the behavior of internal signals during normal device operation, without requiring extra I/O pins or external lab equipment.

The Signal Tap logic analyzer supports data acquisition in the static and PR regions. Moreover, you can debug multiple personas present in a PR region and multiple PR regions.

Related Information

- Design Debugging with the Signal Tap Logic Analyzer in *Intel Quartus Prime Pro Edition User Guide: Debug Tools*
- Creating a Partial Reconfiguration Design in *Intel Quartus Prime Pro Edition User Guide: Partial Reconfiguration*
- AN 953: Partially Reconfiguring a Design on an Intel Agilex F-Series FPGA Development Board

1.1. PR Debug Considerations

Debugging a PR design requires planning. Before compiling, you must decide whether to tap signals in the static region, which PR region you want to debug, and which personas in the PR region you want to debug.

If you have multiple personas in your design, you must also instantiate the Intel Configuration Reset Release Endpoint to Debug Logic IP in each PR region. This IP ensures proper function by providing a reset signal to debug logic, such as Signal Tap, after partial reconfiguration. This reset signal must be high during configuration, and then this reset signal must go low once partial reconfiguration is complete. You must not release this reset signal after releasing the PR logic reset. The time of this reset release affects the Signal Tap power-up trigger feature. The reset signal must stay low until the next reconfiguration.

Note: Do not assert this reset input while the device is in the user operational mode. Asserting this reset input while the device is in the user operational mode results in incorrect operation in Signal Tap and other debugging tools.

If you omit the Intel Configuration Reset Release Endpoint to Debug Logic IP from your PR design, The Compiler issues the following error message:

```
Error(11176): Alt_sld_fab_1.alt_sld_fab_1.alt_sld_fab_1: The Intel
Configuration Reset Release
Endpoint to Debug Logic IP must be instantiated to provide the reset signal to
the debug logic,
such as Signal Tap, etc. after the partial configuration is performed.
```

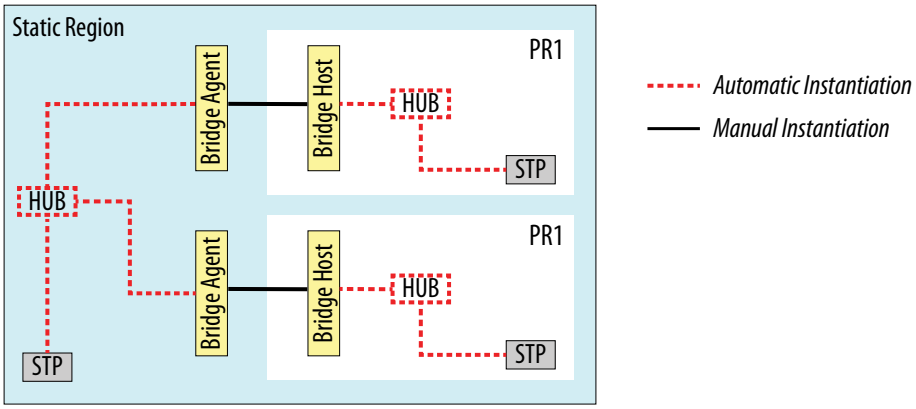
To ensure visibility, the debugging fabric must extend to all the regions that you want to tap. The Intel Quartus Prime software enables you to extend the debug fabric by using debug bridge components: the SLD JTAG Bridge Agent Intel FPGA IP and the SLD JTAG Bridge Host Intel FPGA IP.

To incorporate these components to the design, for each PR region in the design that you want to debug:

1. Instantiate the SLD JTAG Bridge Agent Intel FPGA IP in the static region.
2. Instantiate the SLD JTAG Bridge Host Intel FPGA IP and the Intel Configuration Reset Release Endpoint to Debug Logic in the PR region of the default persona.
3. Instantiate the SLD JTAG Bridge Host Intel FPGA IP and the Intel Configuration Reset Release Endpoint to Debug Logic on the implementation revisions that you want to debug.

Figure 1. Debug Fabric in PR Design with Signal Tap

The figure shows in solid outline the entities that you instantiate manually, and in dashed outline the entities that the Compiler instantiates automatically.



SLD JTAG Bridge Index

The index is an attribute of the SLD JTAG Bridge Agent that uniquely identifies bridge agents present in the design. You can find information regarding the bridge index in the synthesis report (`<base revision>.syn.rpt`), by looking under **JTAG Bridge Agent Instance Information**. The bridge index for the root partition is always **None**.

Related Information

- Debugging Partial Reconfiguration Designs Using Signal Tap Logic Analyzer in *Intel Quartus Prime Pro Edition User Guide: Debug Tools*
- Debugging PR Designs with the Signal Tap Logic Analyzer in *Intel Quartus Prime Pro Edition User Guide: Partial Reconfiguration*

1.2. Tutorial Software and Hardware Requirements

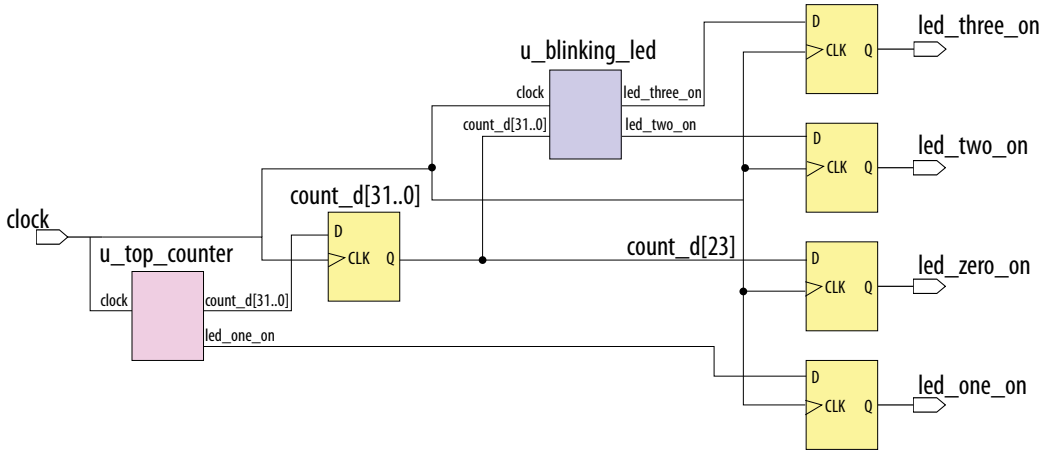
To perform this tutorial, you need the following software and hardware:

- The Intel Quartus Prime Pro Edition software version 21.4 or later. The software includes the Signal Tap Logic Analyzer and the Programmer.
- Intel Agilex F-Series FPGA development kit, or a design board with JTAG connection to the device under test.
- Intel FPGA Download Cable, for communication between the device and the Intel Quartus Prime software.

1.3. Tutorial Design Description

The design for this tutorial consists of one 32-bit counter. At the board level, the design connects the clock to a 50MHz source, and connects the output to four LEDs on the FPGA. Selecting the output from the counter bits in a specific sequence causes the LEDs to blink at a specific frequency.

Figure 2. Flat Reference Design without PR Partitioning



1.4. Downloading the Tutorial Design

The partial reconfiguration tutorial files are available in:

https://github.com/intel/fpga-partial-reconfig

To download the tutorial:

1. In the web page, click **Code** and then select **Clone** or **Download ZIP**.
2. Unzip the `fpga-partial-reconfig-master.zip` file.
3. Navigate to the `tutorials/agilex_pcie_devkit_blinking_led_stp` sub-folder to access the design.

1.4.1. Tutorial Design Files

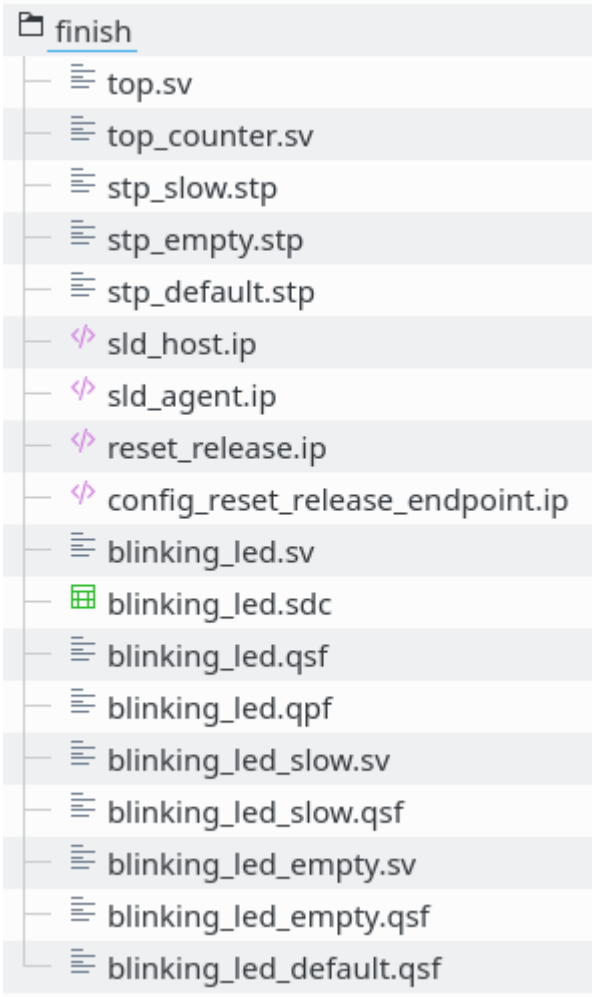
The design folder contains two subfolders: The `start` folder contains the files that you need to follow this tutorial, and the `finish` folder contains the complete set of files you create using this application note. Reference these files at any point during the walkthrough.

Table 1. Description of Tutorial Design Files in start Folder

File Name	Description
`top.sv`	Top-level file. Contains the flat implementation of the design. This module instantiates the `blinking_led` sub-partition and the `top_counter` module. This module also instantiates the SLD JTAG Bridge Agent for debugging purposes.
`top_counter.sv`	Top-level 32-bit counter that controls `LED[1]` directly. The registered output of the counter controls `LED[0]`, and powers `LED[2]` and `LED[3]` via the `blinking_led` module.
`blinking_led.sdc`	Defines the timing constraints for the project.
`blinking_led.sv`	This module acts as the PR partition. The module receives the registered output of `top_counter` module, which controls `LED[2]` and `LED[3]`.This module also instantiates the SLD JTAG Bridge Host and the Intel Configuration Reset Release Endpoint to Debug Logic IP for debugging the default persona.
`blinking_led.qpf`	Intel Quartus Prime project file that contains a list of all the revisions in the project.
`blinking_led.qsf`	Intel Quartus Prime settings file that contains assignments and settings for the base revision of the project.
`blinking_led_default.qsf`	Contains assignments and settings for the `blinking_led_default` implementation revision of the project.
`blinking_led_slow.qsf`	Contains assignments and settings for the `blinking_led_slow` implementation revision of the project.
`blinking_led_empty.qsf`	Contains assignments and settings for the blinking led empty of the project.
`blinking_led_slow.sv`	Slower version of the PR logic. On this version, the led blinks at a slower rate than the default PR persona. The module receives the registered output of `top_counter` module, which controls `LED[2]`and `LED[3]`. This module also instantiates the SLD JTAG Bridge Host and the Intel Configuration Reset Release Endpoint to Debug Logic IP for debugging this persona.
`blinking_led_empty.sv`	Empty version of the PR logic. This module holds the outputs at a constant. The module receives the registered output of `top_counter` module, which controls `LED[2]` and `LED[3]`. This module also instantiates the SLD JTAG Bridge Host and the Intel Configuration Reset Release Endpoint to Debug Logic IP for debugging this persona.

The following Figure shows the list of files in the `finish` folder:

Figure 3. Tutorial Design Files in finish Folder



2. Tutorial Walkthrough

This tutorial describes preparing the blinking_led design for debug with the Signal Tap Logic Analyzer.

This Application Note does not describe turning a non-PR design to a PR design. Refer to *AN 953: Partially Reconfiguring a Design: on an Intel Agilex F-Series FPGA Development Board* for examples of the following tasks:

- Creating a design partition
- Allocating placement and routing regions for a PR partition
- Defining personas
- Creating project revisions
- Preparing PR implementation revisions

Process Description

To tap signals in a PR design, you extend the debug fabric to the PR regions when creating the base revision, and then define debug components for the implementation revisions.

Tutorial Steps

This tutorial includes the following steps:

Related Information

AN 953: Partially Reconfiguring a Design on an Intel Agilex F-Series FPGA Development Board

2.1. Step 1: Getting Started

Copy the reference design files to your working environment and compile the initial design for this tutorial:

1. Before you begin, download the tutorial files.
2. In your working environment, create a directory named `agilex_pcie_devkit_blinking_led_stp`.
3. Copy the downloaded `tutorials/agilex_pcie_devkit_blinking_led_stp/start` sub-folder to your working directory.
4. In the Intel Quartus Prime Pro Edition software, click **File ➤ Open Project** and select `blinking_led.qpf`.
5. Click **Project ➤ Revisions** and set `blinking_led` as the current revision.
6. Click **Processing ➤ Start Compilation**.
7. Repeat the following steps to complete Analysis and Synthesis for the `blinking_led_slow`, `blinking_led_default`, and `blinking_led_empty` revisions:
   a. Change the current revision by clicking **Project ➤ Revisions** and selecting a revision to set as the current revision.
   b. Click **Processing ➤ Start ➤ Start Analysis and Synthesis**.

Related Information

Downloading the Tutorial Design on page 5

2.2. Step 2: Preparing the Base Revision

To prepare the base revision, extend the debug fabric to the PR regions that you want to debug.

To extend the debug fabric to the PR regions that you want to debug:

1. Instantiate the SLD JTAG Bridge Agent in the static region
2. Instantiate the SLD JTAG Bridge Host in the default persona of the PR region

For the debug logic to be function properly after partial reconfiguration, the design needs a reset signal. To add a reset signal to the design:

1. Instantiate the Reset Release Intel FPGA IP in the static region
2. Instantiate the Intel Configuration Reset Release Endpoint to Debug Logic IP in the PR region.

The Reset Release Intel FPGA IP generates the reset signal for the Configuration Reset Release Endpoint to Debug Logic IP.

2.2.1. Preparing the Static Region

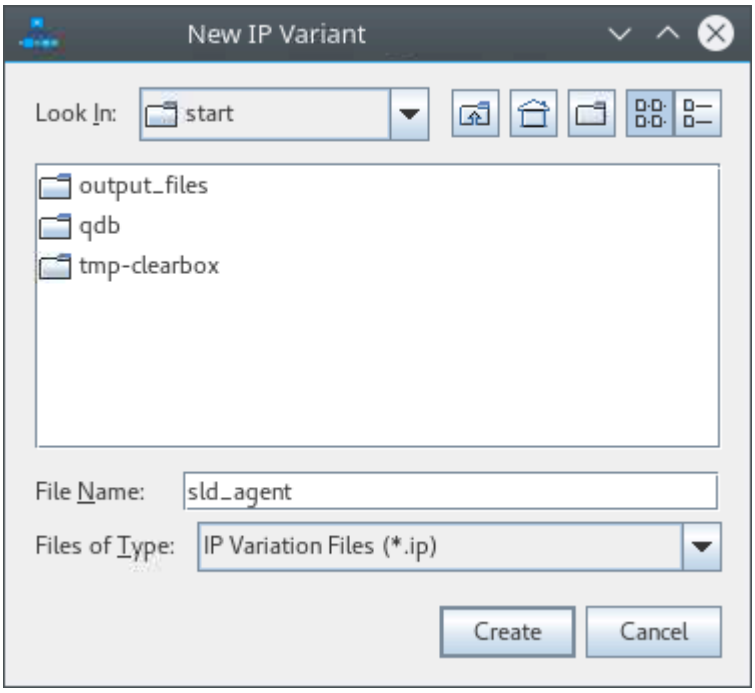
1. Ensure that `blinking_led` is the current revision.

   To change the revision to the `blinking_led` revision, click **Project ➤ Revisions** and set `blinking_led` as the current revision.
2. Add the SLD JTAG Bridge Agent Intel FPGA IP to the design:

a. In the IP Catalog (**Tools ➤ IP Catalog**), type `SLD JTAG Bridge Agent`, and double-click the **SLD JTAG Bridge Agent Intel FPGA IP**.

b. In the **New IP Variant** dialog box, type `sld_agent` as the file name, and then click **Create**.

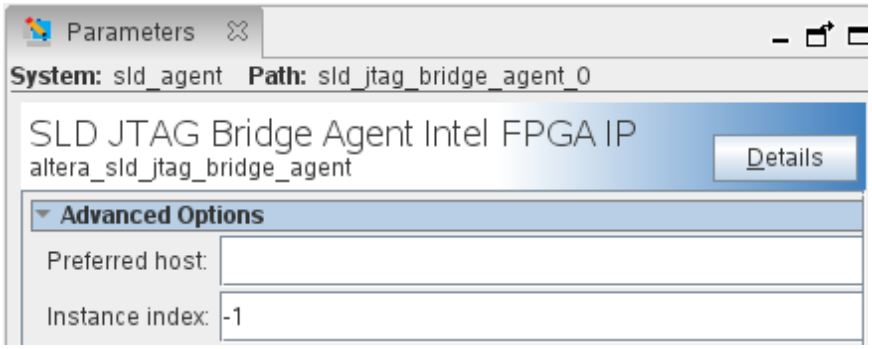
Figure 4. New IP Variant Dialog Box



c. In the parameter editor, use the default parameterization for `sld_agent`. Click **Generate HDL** and then click **Generate**.

Save your changes, if prompted.

Figure 5. SLD JTAG Bridge Agent Intel FPGA IP Parameters

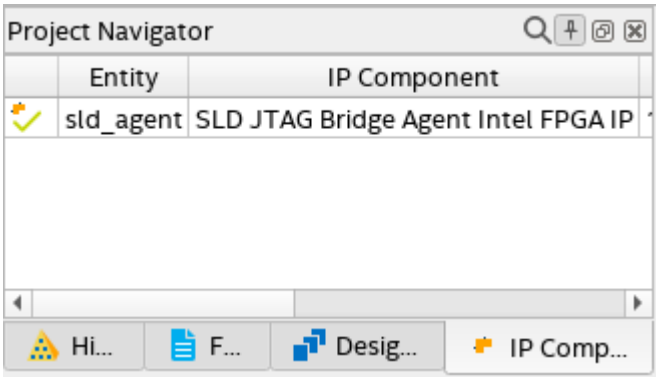


The parameter editor generates the `sld_agent.ip` IP variation file and adds the file to the `blinking_led` project.

d. Close the parameter editor.

e. Verify whether the `sld_agent` IP variant appears in the **IP Components** tab of the Project Navigator.

Figure 6. sld_agent IP Variant in Project Navigator



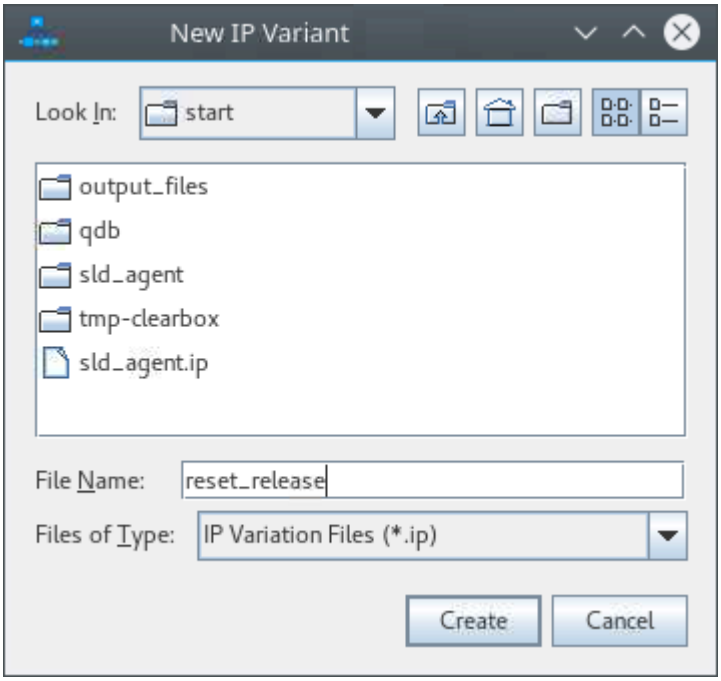
If the IP variant does not appear in the Project Navigator, click **Project ➤ Add/Remove Files in Project**, find the `sld_agent.ip` file, and add to the project.

f. In the `top.sv` file, instantiate the `sld_agent` IP in the base revision by uncommenting the following block of code:

```
//==============
//Uncomment this block to enable Signal Tap

wire tck;
wire tms;
wire tdi;
wire vir_tdi;
wire ena;
wire tdo;

sld_agent u_sld_agent (
 .tck  (tck),  //  output,  width = 1, connect_to_bridge_host.tck
 .tms  (tms),  //  output,  width = 1, .tms
 .tdi  (tdi),  //  output,  width = 1, .tdi
 .vir_tdi (vir_tdi), //  output,  width = 1, .vir_tdi
 .ena  (ena),  //  output,  width = 1,  .ena
 .tdo  (tdo)   //  input,   width = 1,  .tdo
);

//==============
```

3. Add the Reset Release Intel FPGA IP to the design:
   a. In the IP Catalog (**Tools ➤ IP Catalog**), type `Reset Release`, and double-click the **Reset Release Intel FPGA IP**.
   b. In the **New IP Variant** dialog box, type `reset_release` as the file name, and then click **Create**.

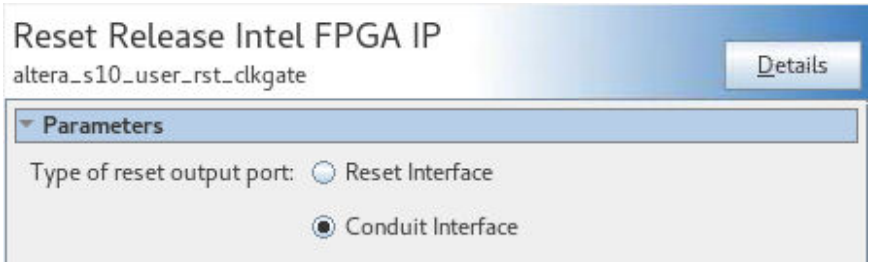
Figure 7. New IP Variant Dialog Box



c. In the parameter editor, under **Type of reset output port**, select **Conduit Interface**. Click **Generate HDL** and then click **Generate**.

Save your changes, if prompted.

Figure 8. Reset Release Intel FPGA IP Parameters

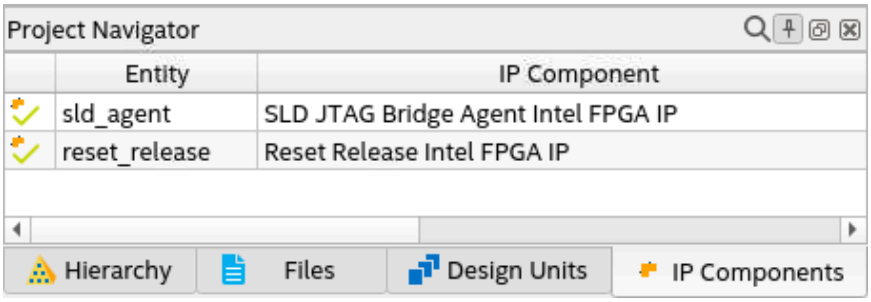


The parameter editor generates the `reset_release.ip` IP variation file and adds the file to the `blinking_led` project.

d. Close the parameter editor.

e. Verify whether the `reset_release` IP variant appears in the **IP Components** tab of the Project Navigator.

Figure 9. reset_release IP Variant in Project Navigator



If the IP variant does not appear in the Project Navigator, click **Project ➤ Add/Remove Files in Project**, find the `reset.ip` file, and add it to the project.

f. In the `blinking_led.sv` file, instantiate the `reset_release` IP in the default persona by uncommenting the following block of code:

```
//==============
//Uncomment this block to enable Signal Tap

wire connect_to_conf_rst;

reset_release u_reset_release (ninit_done (connect_to_conf_rst)  //
output,  width = 1, ninit_done.ninit_done
 );

//==============
```

Related Information

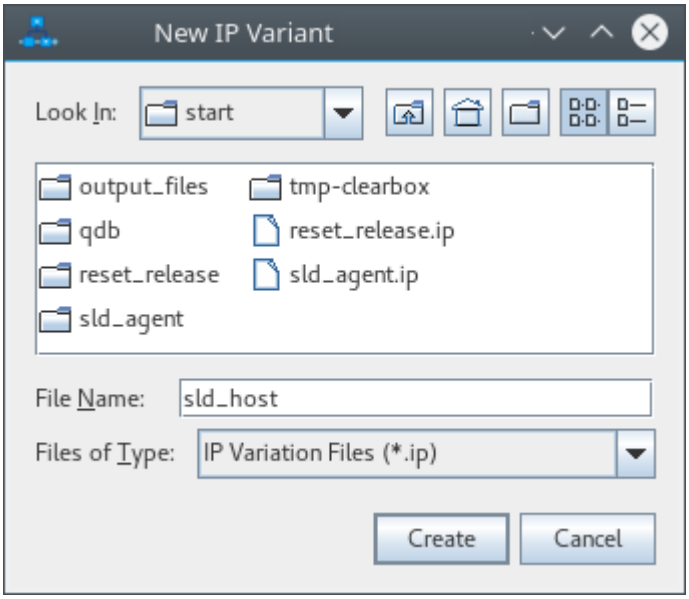
- Instantiating the SLD JTAG Bridge Agent in *Intel Quartus Prime Pro Edition User Guide: Debug Tools*
- Including the Reset Release Intel FPGA IP in Your Design in *Intel Agilex Configuration User Guide*

2.2.2. Preparing the Default PR Persona

In this phase you instantiate the SLD JTAG Bridge Host Intel FPGA IP and the Intel Configuration Reset Release Endpoint to Debug Logic IP in the PR region that you want to debug.

1. Add the SLD JTAG Bridge Host Intel FPGA IP to the design:
   a. In the IP Catalog (**Tools ➤ IP Catalog**), type `SLD JTAG Bridge Host`, and double-click the **SLD JTAG Bridge Host Intel FPGA IP**.
   b. In the **New IP Variant** dialog box, type `sld_host` as the file name, and then click **Create**.

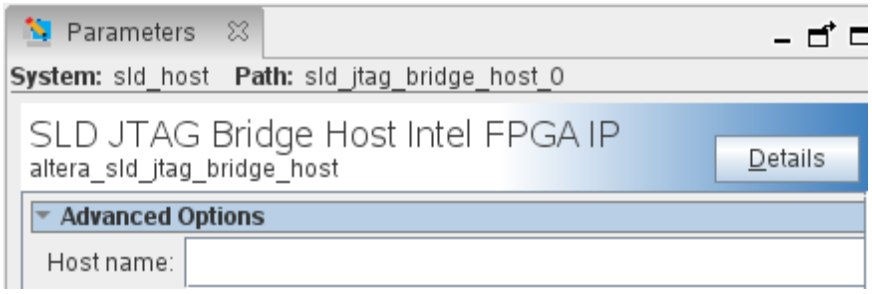
Figure 10. New IP Variant Dialog Box



c. In the parameter editor, keep the default parameterization for `sld_host`. Click **Generate HDL** and then click **Generate**.

   Save your changes, if prompted.

Figure 11. SLD JTAG Bridge Host Intel FPGA IP Parameters

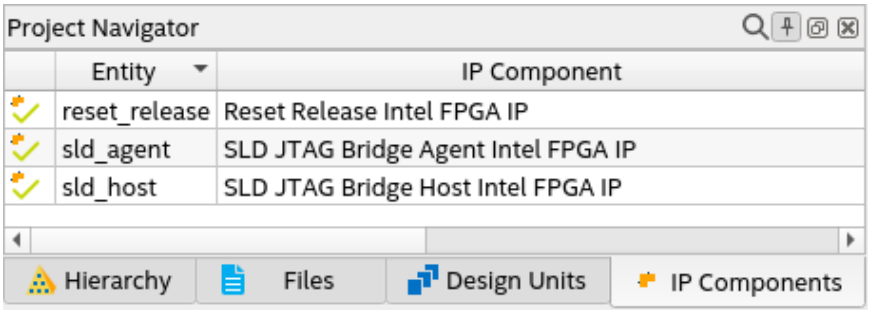


The parameter editor generates the `sld_host.ip` IP variation file and adds the file to the `blinking_led` project.

d. Close the parameter editor.

e. Verify whether the `sld_host` IP variant appears in the **IP Components** tab of the Project Navigator.

Figure 12. sld_host IP Variant in Project Navigator



If the IP variant does not appear in the Project Navigator, click **Project ➤ Add/Remove Files in Project**, find the `sld_host.ip` file, and add it to the project.

f. In the `blinking_led.sv` file, instantiate the `sld_host` IP in the default persona by uncommenting the following block of code:

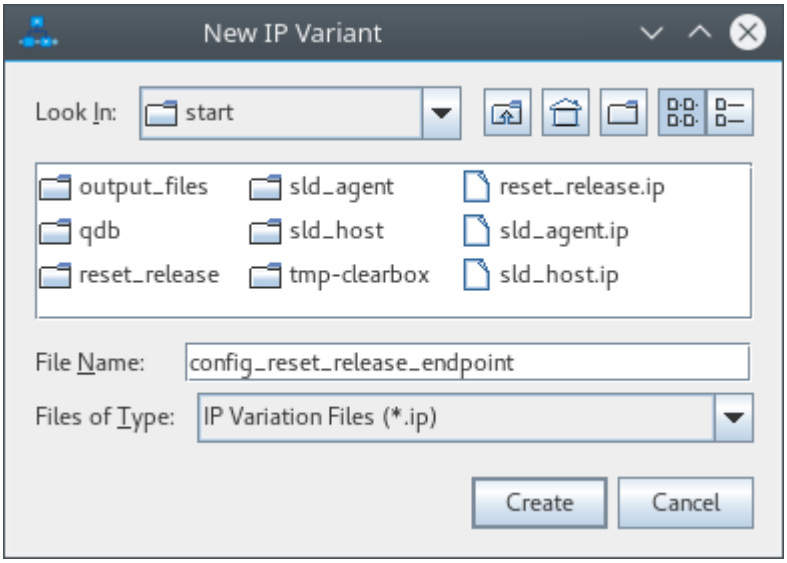
```
//==================
//Uncomment this block to enable Signal Tap

sld_host u_sld_hostled_two_on (
 .tck  (tck),  //  input,  width = 1, connect_to_bridge_host.tck
 .tms  (tms),  //  input,  width = 1, .tms
 .tdi  (tdi),  //  input,  width = 1, .tdi
 .vir_tdi (vir_tdi), //  input,  width = 1, .vir_tdi
 .ena  (ena),  //  input,  width = 1, .ena
 .tdo  (tdo)   //  output, width = 1, .tdo
);

//==================
```

2. Add the Intel Configuration Reset Release Endpoint to Debug Logic IP to the design:

   a. In the IP Catalog (**Tools ➤ IP Catalog**), type `Configuration Reset Release Endpoint`, and double-click the **Intel Configuration Reset Release Endpoint to Debug Logic**.

   b. In the **New IP Variant** dialog box, type `config_reset_release_endpoint` as the file name, and then click **Create**.

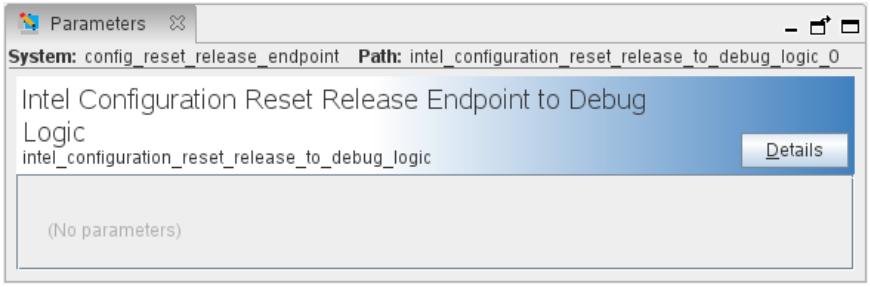
Figure 13. New IP Variant Dialog Box



c. In the parameter editor, keep the default parameterization for `config_reset_release_endpoint`. Click **Generate HDL** and then click **Generate**.

   Save your changes, if prompted.

Figure 14. Intel Configuration Reset Release Endpoint to Debug Logic Parameters

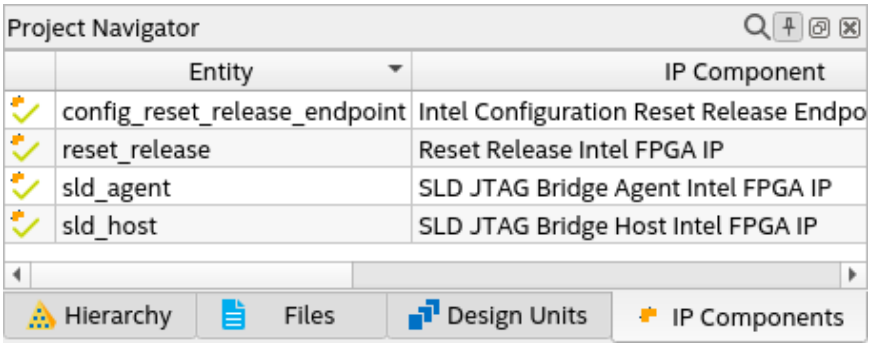


The parameter editor generates the `config_reset_release_endpoint.ip` IP variation file and adds the file to the `blinking_led` project.

d. Close the parameter editor.

e. Verify whether the `config_reset_release_endpoint` IP variant appears in the **IP Components** tab of the Project Navigator.

Figure 15. config_reset_release_endpoint IP Variant in Project Navigator



If the IP variant does not appear in the Project Navigator, click **Project ➤ Add/Remove Files in Project**, find the `config_reset_release_endpoint.ip` file, and add it to the project.

f. In the `blinking_led.sv` file, instantiate the `config_reset_release_endpoint` IP in the default persona by uncommenting the following block of code:

```
//==================
//Uncomment this block to enable Signal Tap

config_reset_release_endpoint u_config_reset_release_endpoint
(.conf_reset (reset)  //   input,  width = 1, conf_reset_in.reset
);

//==================
```

3. In the `blinking_led.sv` file, update the port definition of the default PR persona to include the following ports by uncommenting the following block of code:

```
//====================
// Uncomment this block to enable Signal Tap

input wire reset,
input wire tck,
input wire tms,
input wire tdi,
input wire vir_tdi,
input wire ena,
output wire tdo,

//====================
```

4. In the `top.sv` file, update the instantiation of the persona to include the `sld_host` and `config_reset_release_endpoint` ports by uncommenting the following block of code:

```
//==================
//Uncomment this block to enable Signal Tap

.reset   (connect_to_conf_rst),
.tck    (tck),      //   input,  width = 1, connect_to_bridge_host.tck
.tms    (tms),      //   input,  width = 1,                         .tms
.tdi    (tdi),      //   input,  width = 1,                         .tdi
.vir_tdi (vir_tdi),  //   input,  width = 1,                          .vir_tdi
.ena    (ena),      //   input,  width = 1,                         .ena
```

```
.tdo    (tdo),      //  output,  width = 1,                    .tdo
//====================
```

Related Information

Instantiating the SLD JTAG Bridge Host in *Intel Quartus Prime Pro Edition User Guide: Debug Tools*

2.3. Step 3: Preparing the Implementation Revisions for Debugging

To prepare the implementation revisions for debug, instantiate the SLD JTAG Bridge Host Intel FPGA IP and the Intel Configuration Reset Release Endpoint to Debug Logic IP. Then, add a `.stp` file to the implementation revisions that you want to debug.

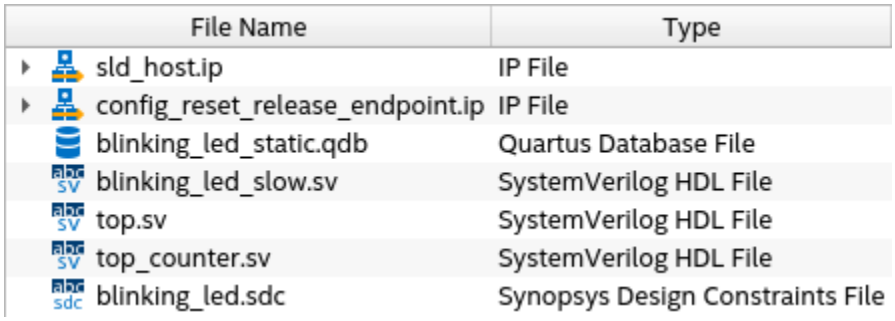
In this step, you prepare each of the revisions for debugging by repeating the preparation procedure for each of the following revisions:

- `blinking_led_slow`
- `blinking_led_default`
- `blinking_led_empty`

To prepare the `blinking_led_slow` revision for debug:

1. In the Intel Quartus Prime GUI, set `blinking_led_slow` as the current revision.
2. Include `sld_host.ip` and `config_reset_release_endpoint.ip` as a project file in the `blinking_led_slow` implementation revision.

Figure 16. Files in `blinking_led_slow` Project Overview After Adding `sld_host.ip` and `config_reset_release_endpoint.ip` Files



3. In the `blinking_led_slow.sv` file, update the port definition for the PR personas to include the required ports by uncommenting this block of code :

```
//=================
//Uncomment this block to enable Signal Tap

input wire reset,
input wire tck,
input wire tms,
input wire tdi,
input wire vir_tdi,
input wire ena,
output wire tdo,

//=================
```

4. In the `blinking_led_slow.sv` file, instantiate the `sld_host` IP in this persona by uncommenting the following block of code:

```
/==================
// Uncomment this block to enable Signal Tap

sld_host u_sld_hostled_two_on (
    .tck    (tck),     //   input,  width = 1, connect_to_bridge_host.tck
    .tms    (tms),     //   input,  width = 1,                         .tms
    .tdi    (tdi),     //   input,  width = 1,                         .tdi
    .vir_tdi (vir_tdi), //   input,  width = 1,                     .vir_tdi
    .ena    (ena),     //   input,  width = 1,                         .ena
    .tdo    (tdo)      //  output,  width = 1,                         .tdo
);

//==================
```

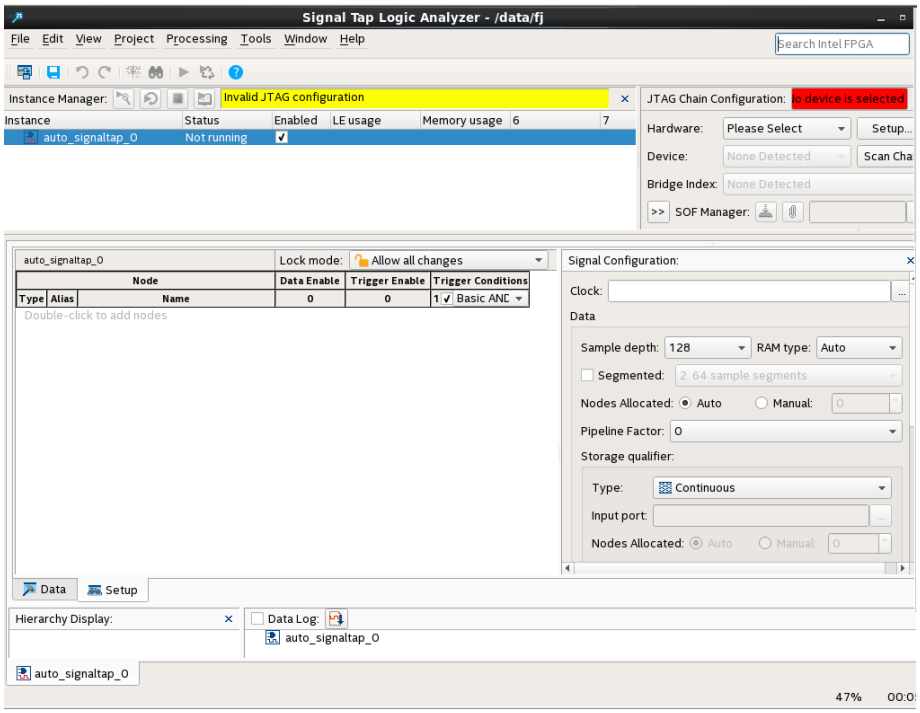
5. In the the `blinking_led_slow.sv` file, instantiate the `config_reset_release_endpoint` IP in this persona by uncommenting the following block of code:

```
//==================
//Uncomment this block to enable Signal Tap

config_reset_release_endpoint u_config_reset_release_endpoint (
    .conf_reset (reset)  //   input,  width = 1, conf_reset_in.reset
);

//==================
```

6. Click **Tools ➤ Signal Tap Logic Analyzer** to open the Signal Tap Logic Analyzer Window.

   If you are prompted to choose a template, select the Default template and click **Create**.

Figure 17. Signal Tap Logic Analyzer Window

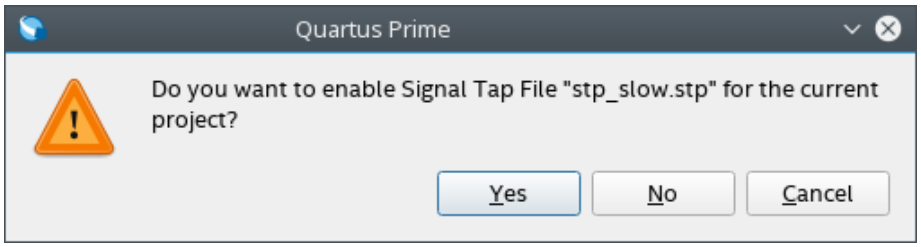


7. Click **File ➤ Save As**, and save the file as `stp_slow.stp`.

   You might get a message saying **Input "Data and Trigger" is empty**. Click **OK** to dismiss the message.

   A dialog box appears prompting you to enable Signal Tap file `stp_slow.stp` for the current project

Figure 18. Enable stp_slow.stp for the Current Project



8. Click **Yes**.

Repeat this procedure for the remaining revisions. Change the name of the Signal Tap file to correspond to the revision name as follows:

Revision	Signal Tap File Name
`blinking_led_default`	`stp_default.stp`
`blinking_led_empty`	`stp_empty.stp`

You can disable Signal Tap in the project by clicking **Assignments ➤ Settings**. In the **Category** pane select Signal Tap Logic Analyzer. Then, turn off **Enable Signal Tap Logic Analyzer**.

Related Information

Tutorial Design Description on page 5

2.4. Step 4: Configuring Signal Tap Logic Analyzer

In this step, you configure Signal Tap Logic Analyzer for each revision.

Revision	Signal Tap File Name
`blinking_led_slow`	`stp_slow.stp`
`blinking_led_default`	`stp_default.stp`
`blinking_led_empty`	`stp_empty.stp`

Repeat the following steps for each revision:

1. Set the current revision in the Intel Quartus Prime GUI.
2. Open the Signal Tap file that corresponds to the set revision.
3. Tap signals in the implementation persona.
4. Configure data acquisition, including the acquisition clock and storage parameters.
5. Set the trigger conditions for recording data.
6. Save your changes before you continue to compiling the design.

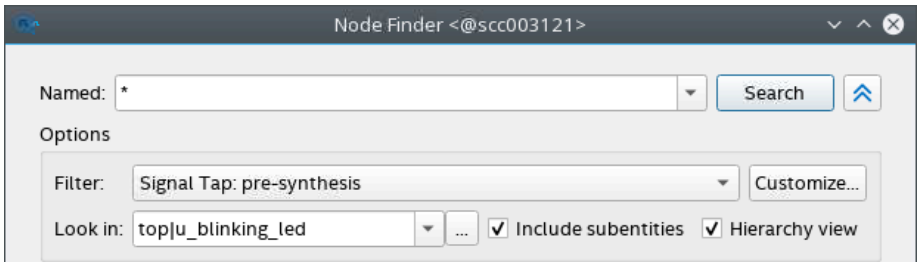
2.4.1. Tapping Signals in the Implementation Persona

To add signals from the implementation persona to the Signal Tap logic analyzer:

1. In the **Setup** tab, double-click anywhere to open the Node Finder.
2. Set the following search fields, and then click **Search**

Field	Value
Named	*
Filter	Signal Tap: pre-synthesis
Look in	`top|u_blinking_led`

Figure 19. Search Parameters to Find Signals



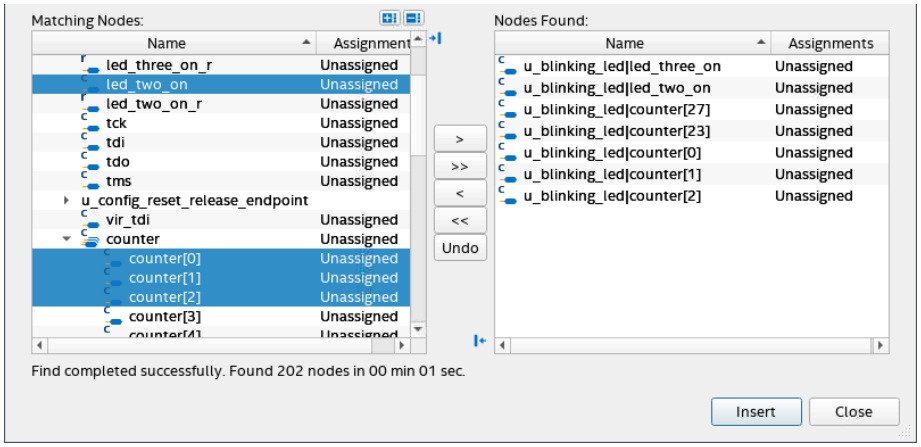
This action displays all the nodes that you can probe in this revision.

3. From the **Matching Nodes** list, select the following nodes and then click **>**:

- led_three_on
- led_two_on
- counter[0]
- counter[1]
- counter[2]
- counter[23]
- counter[27]

This action adds the signals to the **Nodes Found** list.

Figure 20. Signals in Nodes Found List



4. Click **Insert** and then click **Close**.

The signals now appear on the **Data** and **Setup** tabs of the Signal Tap GUI.

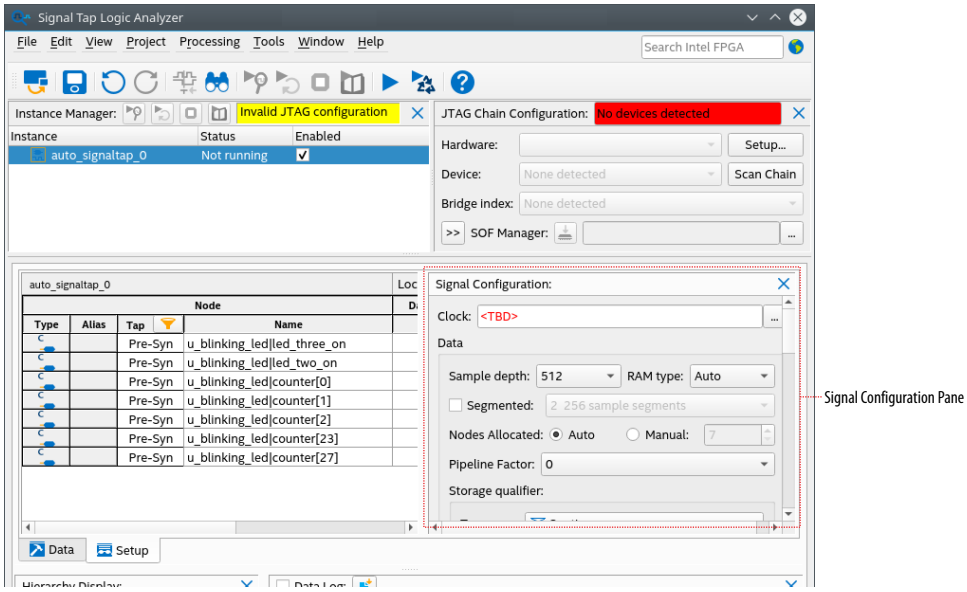
Related Information

Adding Signals to the Signal Tap File in *Intel Quartus Prime Pro Edition User Guide: Debug Tools*

2.4.2. Configuring Data Acquisition

Specify the acquisition parameters in the **Signal Configuration** pane on the **Setup** tab of the Signal Tap Logic Analyzer.

Figure 21. Signal Configuration Pane



2.4.2.1. Add Acquisition Clock

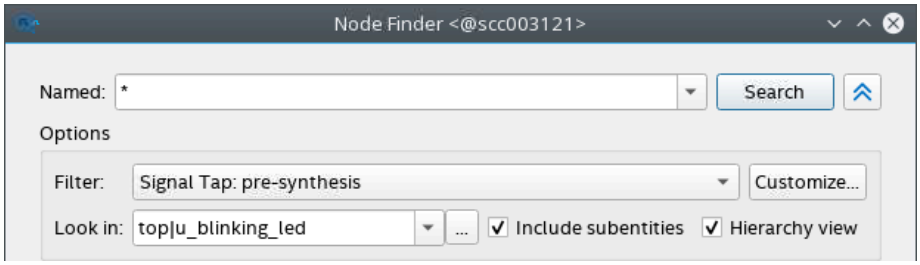
Specify the reference clock that Signal Tap uses during acquisition.

Perform the following steps in the **Signal Configuration** pane:

1. Next to **Clock**, click **...** to open the **Node Finder**.
2. Set the following search parameters:

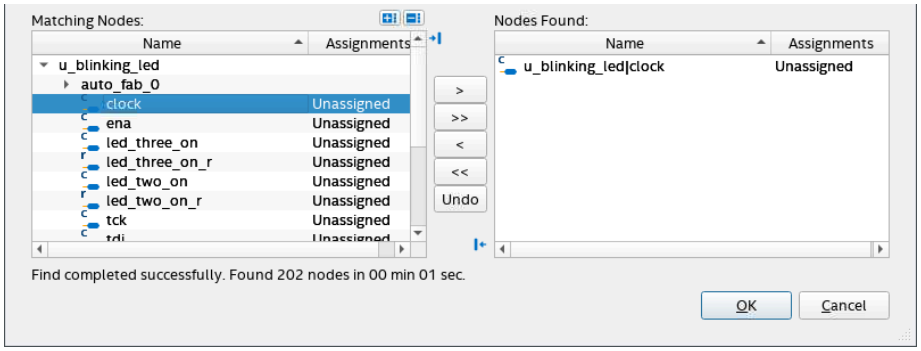
Field	Value
Named	*
Look in	`top|u_blinking_led`

Figure 22. Search Parameters to Find the Clock



3. Click **Search**.

Figure 23. Select Clock in Node Finder



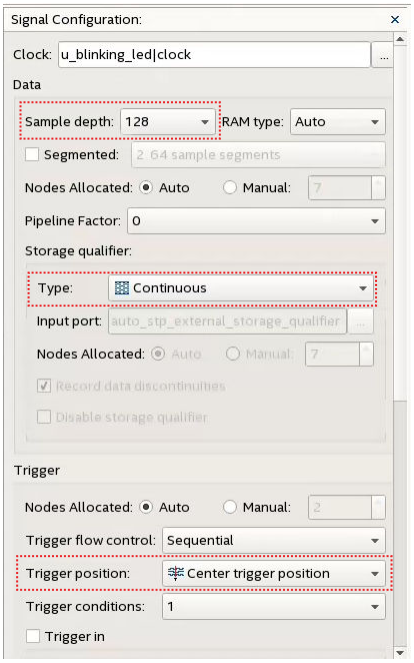
4. Select `clock`, click **>**, and then click **OK**.

2.4.2.2. Add Storage Parameters

Storage parameters define the number of samples the Signal Tap Logic Analyzer captures and stores, how to organize this samples, and the location of the sample with respect to the trigger activation.

1. In **Sample Depth**, select 128.
2. In **Storage Qualifier**, set **Type** as **Continuous**.
3. In **Trigger Position**, select **Center Trigger Position**.

Figure 24. Acquisition Settings for Tutorial



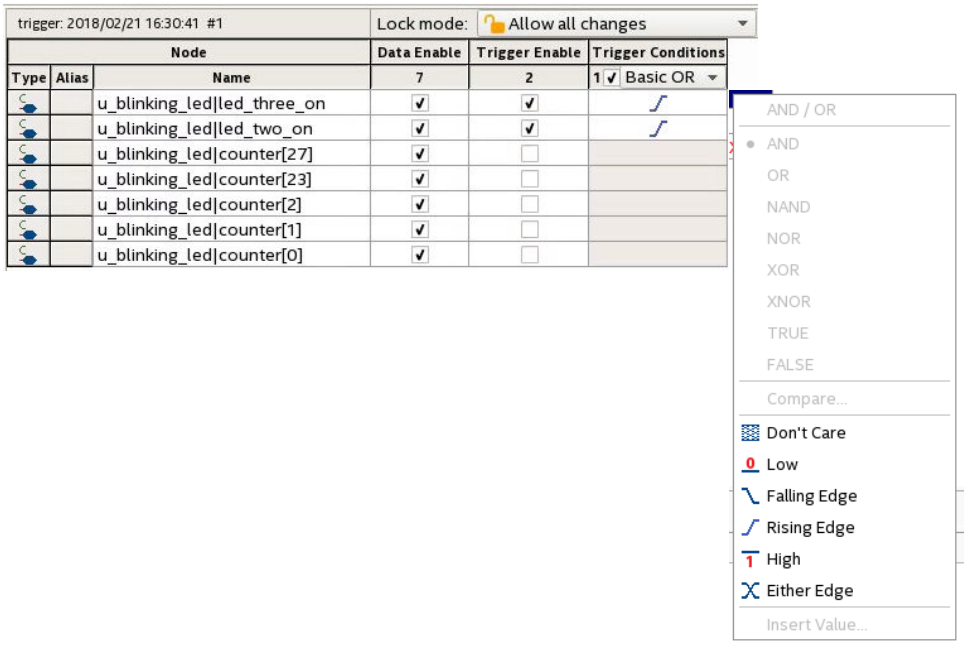
2.4.3. Setting Trigger Conditions

Direct the Signal Tap Logic Analyzer to record data only after `u_blinking_led|led_three_on` or `u_blinking_led|led_two_on` does a rising edge transition:

To set trigger conditions for the current revision:

1. In the **Setup** tab of the Signal Tap Logic Analyzer window, turn on the box under the **Trigger Condition** column
2. Open the drop-down menu and select **Basic OR**.
3. For **u_blinking_led|led_three_on** and **u_blinking_led|led_two_on**, turn on **Trigger Enable** and select **Rising Edge** as the trigger type.
4. For all the other signals, turn off **Trigger Enable**.

Figure 25. Trigger Conditions



Defining trigger conditions completes the Signal Tap instance configuration.

Save your changes before you continue to compiling the design.

2.5. Step 5: Generating Programming Files

The design is now ready for compilation. The Intel Quartus Prime Compiler generates files that you then program into the FPGA. This Partial Reconfiguration design requires generating `.sof` and `.rbf` files.

Compile each revision (`blinking_led`, `blinking_led_slow`, `blinking_led_default`, and `blinking_led_empty`) in the project as follows:

1. Change the current revision by clicking **Project ➤ Revisions** and selecting a revision to set as the current revision.
2. Click **Processing ➤ Start Compilation**.
3. Repeat these steps for each revision.

Alternatively, you can compile the revisions with the following commands:

```
quartus_sh --flow compile blinking_led -c blinking_led
quartus_sh --flow compile blinking_led -c blinking_led_slow
quartus_sh --flow compile blinking_led -c blinking_led_default
quartus_sh --flow compile blinking_led -c blinking_led_empty
```

If the compilation succeeds, the output files are now in the `output_files` directory.

Related Information

- SRAM Object File (.sof) Definition in *Intel Quartus Prime Pro Edition Help*
- Raw Binary File (.rbf) Definition in *Intel Quartus Prime Pro Edition Help*
- Step 7: Preparing PR Implementation Revisions in *AN 953: Partially Reconfiguring a Design: on an Intel Agilex F-Series FPGA Development Board*

2.6. Step 6: Programming the FPGA Device

Before you begin:

1. Connect the power supply to the Intel Agilex F-Series FPGA development board.
2. Connect the USB Blaster cable between your PC USB port and the USB Blaster port on the development board.

Note: This tutorial utilizes the Intel Agilex F-Series FPGA development board on the bench, outside of the PCIe* slot in your host machine.

To program the design on the board:

1. In the Intel Quartus Prime software, click **Tools ➤ Programmer**.
2. In the Programmer, click **Hardware Setup** and select **USB-Blaster**
3. Click **Auto Detect** and select the appropriate Intel Agilex device for your development board.
4. Click **OK**. The Intel Quartus Prime software detects and updates the Programmer with the FPGA devices on the board.
5. Select the appropriate Intel Agilex device for your development board and click **Change File** and load the `blinking_led.sof` file.
6. Enable **Program/Configure** for `blinking_led.sof` file.
7. Click **Start** and wait for the progress bar to reach 100%.
8. To program the PR persona that you want to debug, right-click the `blinking_led.sof` file in the Programmer, and click **Add PR Programming File**.
9. Select the `blinking_led_slow.pr_partition.rbf` file.

10. Disable **Program/Configure** for the `blinking_led.sof` file.
11. Enable **Program/Configure** for `blinking_led_slow.pr_partition.rbf` file and click **Start**.
12. On the board, verify that two of the LEDs are blinking slower than the other two.

Related Information

Troubleshooting PR Programming Errors in *Intel Quartus Prime Pro Edition User Guide: Partial Reconfiguration*

2.7. Step 7: Performing Data Acquisition

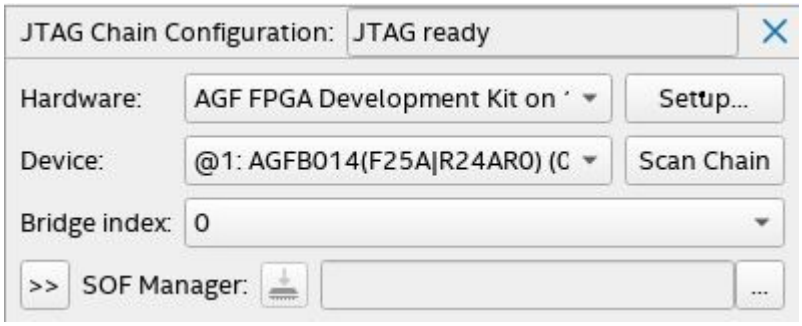
After loading the appropriate `.rbf` onto the board, start data acquisition on the Signal Tap logic analyzer.

To perform data acquisition:

1. Make sure that the Signal Tap Logic Analyzer loads the `.stp` file in the current active revision.
2. In the top right corner of the Signal Tap window, set up the JTAG connection to the board with the following options:

Option	Description
Hardware	**USB-BlasterII**
Device	**AGFB014**
Bridge Index	**0** Bridge index is set to 0 for tapping signals in the PR region.

Figure 26. JTAG Configuration



3. On the Signal Tap toolbar, click **Run Analysis** .

   The analysis may take a few minutes.

   When the analysis finishes, the Signal Tap Logic Analyzer loads the waveforms to the window.

The following section displays the resultant waveforms for all PR configurations.

3. Tutorial Results

By looking at the data acquisition waveform, you can verify whether the signals behave consistently with your expectations. As a reference, the waveforms include counter[2:0] signals.

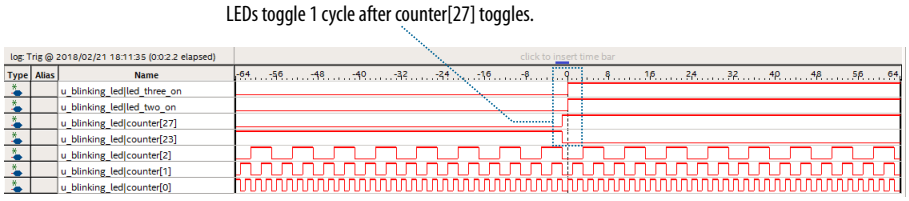
Related Information

Tutorial Design Description on page 5

3.1. Waveforms for Slow Implementation

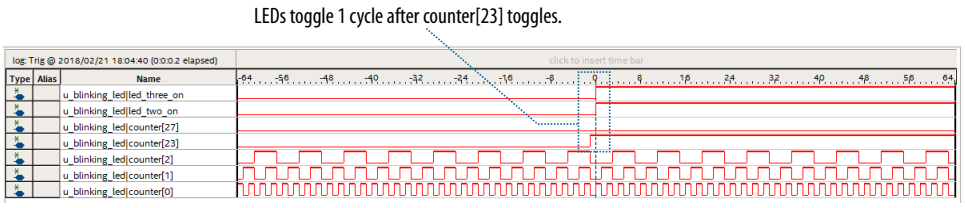
In the Figure, signals `led_three_on` and `led_two_on` show a rising edge one clock cycle after `counter[27]` has a rising edge.

Figure 27. Slow Implementation



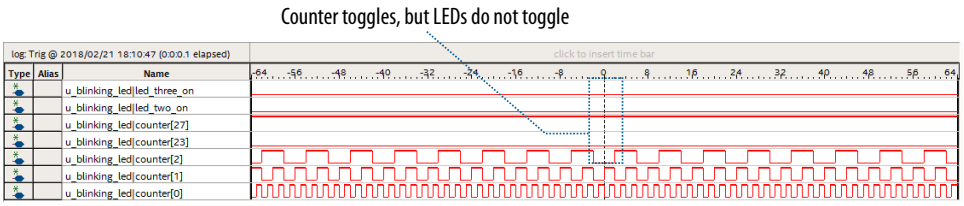
3.2. Waveforms for Default Implementation

Figure 28. Default Implementation



3.3. Waveforms for Empty Implementation

Figure 29. Empty Implementation



4. AN 964: Signal Tap Tutorial for Intel Agilex Partial Reconfiguration Design Archives

For the latest and previous versions of this application note, go to the following URL: https://www.intel.com/content/www/us/en/docs/programmable/710463/.

If a software version is not listed there, the application note for the previous software version applies.