Efinity® Debugger Tutorial

UG-EFN-TUTDBG-v1.1
December 2022
www.efinixinc.com

Contents

Introduction

The Efinity® software includes a hardware Debugger to probe signals in your FPGA design via the JTAG interface. The Debugger includes two debug cores:

- You use a manual flow and the Profile Editor to configure Virtual I/O (vio) cores.
- You can use a manual flow or the Debug Wizard's automated flow to configure Logic Analyzer (la) cores.

The following sections walk you through the Debugger's automated and manual flows.

Note: The Debugger tutorials require the Trion® T20 BGA256 Development Board and Efinity® software v2019.3 or higher. These tutorials assume that you have already installed the Efinity® software and USB driver for the board.

Automated Debugging Flow

This tutorial walks you through the Debugger automated flow using an example **helloworld** design. This tutorial uses the Trion® T20 BGA256 Development Board, the GTKWave waveform viewer, and assumes that you have working knowledge of the Efinity® software.

You add a Logic Analyzer debug core and configure it using the Debug Wizard.

Prepare the Tutorial Files

In this step you set up your environment and copy the Debugger tutorial design to your working directory.

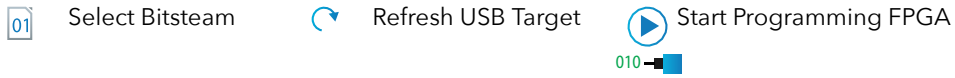
1. Run the Efinity setup script if you have not already done so:
   - Linux: `source <Efinity path>/bin/setup.sh`
   - Windows: `<Efinity path>/bin/setup.bat`
2. Copy the folder *<Efinity path>***/debugger/demo/helloworld-dbg** to your working directory.
3. Connect the Trion® T20 BGA256 Development Board to your computer using a USB cable.

Create a Debug Profile

In this task you add the Logic Analyzer debug core and configure it.

1. Open the **helloworld** project in the **helloworld-dbg** directory.
2. Synthesize the design. You do not need to do a full compile; the Debug Wizard only uses the post-map netlist.
3. Click the Debug Wizard icon in the main icon bar to launch it.
4. In the **Signals from** list, choose **Elaborated Netlist** to browse for signals in the pre-map netlist, or **Post-Map** to use signals from the post-map netlist.
5. Select the `led` and `counter` buses from the list on the left and use the **>>** button to move them to the right. Leave the **Probe Type** at the default, which is **DATA AND TRIGGER**.
6. Click **Next**. The wizard generates a debug profile.
7. Leave **Enable "Auto Instantiation"** turned on. This option enables the debug profile in your project. Click **Finish**.
8. The software prompts you to recompile. Click **OK**.
9. Perform a full compile.

Program the T20 FPGA



You program the Trion® T20 FPGA on the development board using these steps:

1. Choose **Tools > Open Debugger** to launch the Debugger. The programming controls are in the Program box.
2. The Trion T20 Development Board displays as the **USB Target**. If it does not, make sure that the board is connected to your computer and click **Refresh USB Targets**.
3. Click the **Select Image File** button.
4. Browse to the **outflow** directory and choose <*helloworld*>**.bit**.
5. Click **Start Programming**. The console displays programming messages.

Run the Debugger

 Connect Debugger  Disconnect Debugger  Add Net

After you program the FPGA with the design containing the debug core, you can run the Debugger to observe the values on the probed signals. In the Debugger:

1. Click Connect Debugger.
2. In the **Trigger Setup** tab, click Add Net.
3. Select **led[7:0]** and click **OK**.
4. Specify a **Value** of `00001111`, which triggers when the LED output is 00001111.
5. Click **Run**. The Debugger waits for the trigger and then captures data.
6. When the Debugger finishes, it automatically opens the waveform in GTKWave
7. Click Disconnect Debugger to stop the Debugger.

Manual Debugging Flow

This tutorial walks you through the Debugger manual flow using an example **helloworld** design. This tutorial uses the Trion® T20 BGA256 Development Board, the GTKWave waveform viewer, and assumes that you have working knowledge of the Efinity® software.

You add Logic Analyzer and Virtual I/O cores manually in the Debugger's Profile Editor prespective.

Prepare the Tutorial Files

In this step you set up your environment and copy the Debugger tutorial design to your working directory.

1. Run the Efinity setup script if you have not already done so:
   - Linux: `source <Efinity path>/bin/setup.sh`
   - Windows: `<Efinity path>/bin/setup.bat`
2. If you have not already done so, copy the folder **<*Efinity path*>/debugger/demo/helloworld-dbg** to your working directory.
3. Connect the Trion® T20 BGA256 Development Board to your computer using a USB cable.

Create a Debug Profile

 Add Debug Core  Add Probe  Add Source

In this task you add Virtual I/O and Logic Analyzer debug cores to a profile and configure them:

- You add the input signal(s) to control and the output signal(s) to observe to the Virtual I/O core.
- You add the wire, register, or signal to observe to Logic Analyzer core.

Remember to map the clock source after you instantiate the debug core.

1. Open the **helloworld** project in the **helloworld-dbg** directory.
2. Choose **Tools > Open Debugger** to launch the Debugger. Because your project does not have a debug profile, the Debugger opens to the Profile Editor perspective.
3. Click **Add Debug Core > Virtual I/O** to add a new core with the default **Core name** (**vio0**)..
4. Add three probes (in) and three sources (out) with the following name and width settings (leave the other settings at the defaults):

Type	Name	Width	Description
Probe	counter	26	Shows the values for counter[25:0].
Probe	raddr	4	Shows the values for raddr[3:0].
Probe	led	8	Shows the values for the pattern on led[7:0].
Source	vio_reverse	1	Control the reverse button.
Source	vio_mux_sel	1	Control the multipleser select.
Source	vio_maddr	4	Control the memory address maddr[3:0].

5. Click **Add Debug Core > Logic Analyzer** to add a second core with the default **Core name** (**la0**)..
6. Add three probes (in) with the same name and width settings as the probes in the VIO core (leave the other settings at the defaults):

Type	Name	Width	Description
Probe	counter	26	Captures values in counter[25:0]
Probe	raddr	4	Captures the values in maddr[3:0]
Probe	led	8	Captures the values in led[7:0]

7. Click **Generate Debug RTL**. The Debugger creates the file **debug_top.v** and template files (**debug_TEM PLATE.v** and **debug_TEMPLATE.vhd**) in your project directory.
8. Open the **debug_top.v** file and rename `edb_top` as `edb_top_manual`.

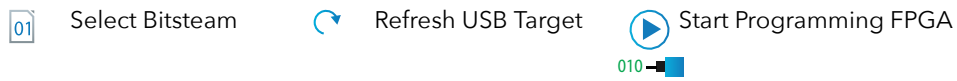
9. Close the Debugger.

Add Debug Code to Your Project

When you generate the debug code, the software copies the **debug_top.v** file to your project directory. You need to add the file to your project, instantiate the RTL, and compile.

1. In the Efinity® main window, click the Project tab under the dashboard.
2. Right-click **Design** and choose **Add**.
3. Browse to your project directory.
4. Select the **debug_top.v** and click **Open**.
5. Add the JTAG User Tap block to the interface design.
   a. Open the Interface Designer.
   b. Select **JTAG User Tap**.
   c. Click Add Block.
   d. Choose **JTAG_USER1** as the **JTAG Resource**.
   e. Generate SDC constraints.
   f. Close the Interface Designer.
6. Edit the **helloworld.v** design to enable the debug code:
   a. Uncomment lines 18 - 28, these lines enable the JTAG USER TAP interface ports.
   b. Uncomment lines 47 - 49, these lines declare the output probe signals and widths.
   c. Comment out line 76.
   d. Uncomment line 77. Steps c and d change the hardware pushbutton to the `vio0` source control signal
   e. Comment line 84.
   f. Uncomment line 85. Steps e and f change the hardware [ushbutton to the `vio0` source control signal.
   g. Uncomment lines 98 - 121, which instantiate the `debug_top` module (this module is in the debug template file).
   h. Save.
7. Compile the design.

Program the T20 FPGA



You program the Trion® T20 FPGA on the development board using these steps:

1. Choose **Tools > Open Debugger** to launch the Debugger. The programming controls are in the Program box.
2. The Trion T20 Development Board displays as the **USB Target**. If it does not, make sure that the board is connected to your computer and click **Refresh USB Targets**.
3. Click the **Select Image File** button.
4. Browse to the **outflow** directory and choose *<helloworld>*.**bit**.
5. Click **Start Programming**. The console displays programming messages.

Run the Debugger

 Connect Debugger  Disconnect Debugger  Add Trigger Condition

After you program the FPGA with the design containing the debug core, you can run the Debugger to observe the values on the probed signals. In the Debugger:

1. Click Connect Debugger. The view opens to the **la0** tab.
2. In the **Trigger Setup** tab, click Add Trigger Condition.
3. Choose **led[7:0]** and click **OK**.
4. Specify a value of `00001111`, which triggers when the LED output is 00001111.
5. Click the **vio0** tab. The **Value** fields show the data captured on the probes.
   To reverse the LED blinking direction, change the **Value** for **vio_reverse** to **1** and press Enter. To stop the LEDs blinking, change the **Value** for **vio_mux_sel** to **1** and press Enter.
6. Click Disconnect Debugger to stop capturing data.

Where to Learn More

The Efinity® software includes documentation as PDF user guides and on-line HTML help. This documentation is provided with the software. You can also access the latest versions of PDF documentation in the Support Center:

- Efinity Software User Guide
- Efinity Synthesis User Guide
- Efinity Timing Closure User Guide
- Efinity Software Installation User Guide
- Efinity Trion Tutorial
- Efinity Debugger Tutorial
- Titanium Interfaces User Guide
- Trion Interfaces User Guide
- Efinity Interface Designer Python API
- Quantum Trion Primitives User Guide
- Quantum Titanium Primitives User Guide

In addition to documentation, Efinix field application engineers have created a series of videos to help you learn about aspects of the software. You can view these videos in the Support Center.

Revision History

Table 1: Revision History

Date	Version	Description
December 2022	1.1	Corrected the source and probe names for the manual flow's Run the Debugger topic.
August 2022	1.0	Initial release. The content in this tutorial originally appeared in the Efinity Trion Tutorial.