Efinity® Trion® Tutorial

UG-EFN-TUTORIAL-v5.0
December 2019
www.efinixinc.com

Contents

Introduction

This tutorial walks you through the Efinix Efinity® software flow from beginning to end using the example project helloworld. In addition to running the flow, you will adjust design constraints. You will use both the graphical user interface (GUI) and the command-line interface.

Learn more: For detailed information on the software functionality, refer to the Efinity Software User Guide.

Hardware & Software Requirements

- Computer with a 64 bit operating system, dual-core processor, and 16 GB RAM
- Efinix Trion® development board
- Linux environments:
  - Operating system:
    - Ubuntu x86-64 v14.04 or later
    - Red Hat Enterprise x86-64 v6 or later
    - CentOS x86-64 v6 or later
  - Linux X11 windowing system (for Efinity® GUI)
  - Udev device manager for Efinix USB programming cable (see Install the USB Driver on page 15)
- Windows environments:
  - Windows 7 or later, 64 bit
  - Microsoft Visual C++ 2015 x64 runtime library
- Your preferred text editor
- (Optional) free Icarus Verilog (iVerilog) simulator, download from iverilog.icarus.com
- (Optional) GTKWave waveform viewer, download from gtkwave.sourceforge.net

Note: You can download the Microsoft library from www.microsoft.com/en-us/download/details.aspx?id=52685.

1.0 Prepare the Tutorial Files

To prepare the tutorial files and set up your environment, perform these steps:

1. Open a terminal window.
2. Change to the directory in which you installed the Efinity® software, *<installation path>***/efinity/***<version>*.
3. Type the following commands:
   - Linux:

```
> source bin/setup.sh
> cd project
> mkdir tutorial
> cp -r helloworld/ tutorial/
> cd tutorial/helloworld
```

   - Windows:

```
> bin\setup.bat
> cd project
> md tutorial
> xcopy helloworld tutorial\helloworld\
> cd tutorial\helloworld
```

2.0 Create Your Project

In this step, you create a new project based on the helloworld example design.


Open the Efinity GUI


Create new project


Choose a directory for project


Import design and constraint files


Delete file

1. Open the Efinity® GUI.
2. Create a new project (**File > Create project** or click the new project button). The Project Editor opens to the Project tab.
3. Under Location, click the Choose a directory to store project data button.
4. Select the **tutorial/helloworld** directory.
5. Click **Choose**.
6. Type `helloworld` in the **Name** box.
7. Type a project description in the **Description** box. For example, My `helloworld example project`.
8. Choose Trion® as the family.
9. Choose **T8F81** as the device. Keep the default timing model selection.
10. Click the Design tab.
11. Next to the **Design** box, click the Import design files button.
12. Browse to the *<install directory>* **/project/tutorial/helloworld** directory.
13. Choose **All Files** under **File to Import**. This setting imports design and constraint files.
14. Click **Choose**.
15. Enter `helloworld` in the **Top Module** field.
16. Select the **helloworld_tb.v** testbench file and delete it. You will use this file for simulation; it is not a design file.
17. Click **OK**.

The Project pane displays the settings for your new project.

3.0 Run the Flow

In this step you run through the complete software flow, including simulation, using the GUI and command line interfaces.

Note: After you run different parts of the flow, use the Netlist pane to explore your design by showing the design hierarchy, the elaborated design, and the synthesized netlist. (You can only view the synthesized netlist after you have performed synthesis.)

3.1 RTL Simulation

First, perform RTL simulation on the design's source files. The **helloworld** design includes the **helloworld_tb.v** testbench file for simulation.

1. Open a terminal window.
2. Type the command:

   Linux:
   ```
   efx_run.py helloworld.xml --flow rtlsim
   ```
   Windows:
   ```
   efx_run.bat helloworld.xml --flow rtlsim
   ```

By default, the Efinity® software calls the iVerilog simulator.

- Use the `--modelsim` option to target the ModelSim simulator.
- Use the `--ncsim` option to target the NCSim simulator.

The software performs simulation and writes the results to the **helloworld.rtl.simlog** file in the **outflow** directory. Double-click the filename under Simulation in the Dashboard Results tab to open the log file in the Efinity Code Editor.

Note: You can also view the results graphically using the GTKWave application. The **helloworld** design includes commands that dump data for use with GTKWave.

GTKWave is an open-source tool that analyzes post-simulation dumpfiles and displays the results in a graphical interface. It includes a waveform viewer and RTL source code navigator. You can use GTKWave with the iVerilog simulator to analyze and debug your simulation model, or to view any VCD waveform. To use GTK Simulator:

1. Download and install the software from gtkwave.sourceforge.net.

   Note: Linux users can use the following commands:

   ```
   sudo apt-get update
   sudo apt-get install gtkwave
   ```

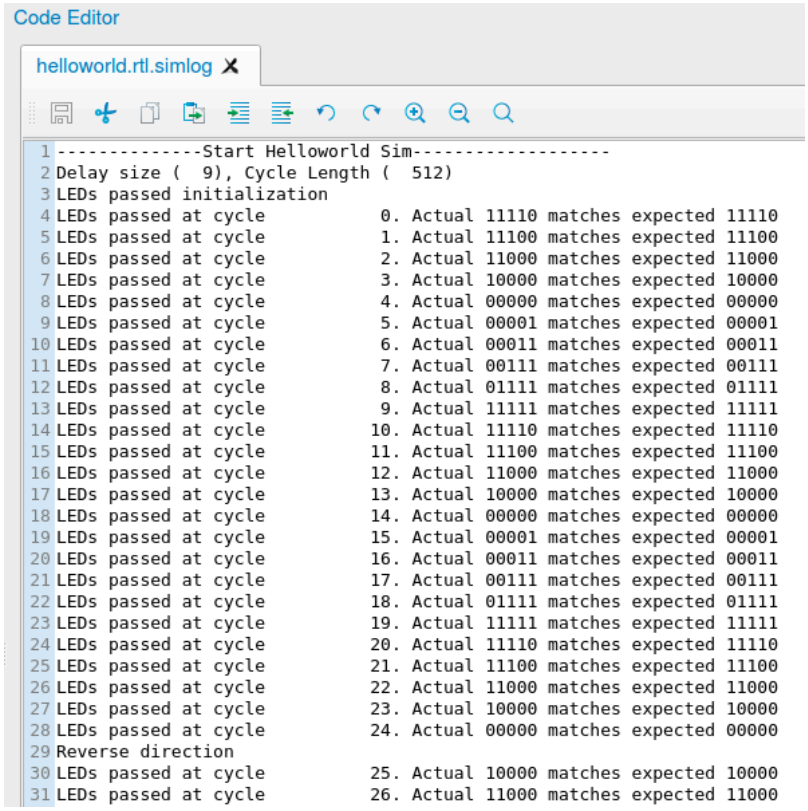
2. Add the following lines to your testbench to generate the dumpfiles:

   ```
   $dumpfile("outflow/<file name>.vcd");
   $dumpvars(0, sim);
   ```

3. Simulate with the iVerilog simulator.
4. Use this command to view the output waveform:

   ```
   gtkwave outflow/<project name>.vcd
   ```

Figure 1: helloworld Simulation Results



3.2 Synthesize the Design

You perform synthesis from the GUI or from the command line.

 Enable/disable automated flow

 Synthesize the design

1. In the GUI Dashboard, turn off the automated flow.
2. Click the Synthesize button.

OR, at the command line, use the command line:

Linux:

```
> efx_run.py helloworld.xml --flow map
```

Windows:

```
> efx_run.bat helloworld.xml --flow map
```

You view the synthesis report files, **helloworld.map.rpt** and **helloworld.map.out**, in the Results pane in the GUI or in the **outflow** directory.

Note: If you run command line operations while the GUI is open, the GUI state does not automatically sync with the command line state.

3.3 Perform Post-Map Simulation

You perform post-map simulation from the command line.

In a terminal, type the command:

Linux:

```
> efx_run.py helloworld.xml --flow mapsim
```

Windows:

```
> efx_run.bat helloworld.xml --flow mapsim
```

By default, the Efinity® software calls the iVerilog simulator.

- Use the `--modelsim` option to target the ModelSim simulator.
- Use the `--ncsim` option to target the NCSim simulator.

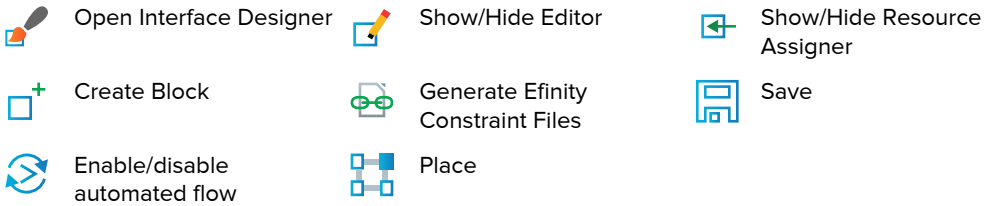
Note: Simulation may take several minutes to run in the iVerilog simulator.

View the report file, **helloworld.map.simlog**, in the Results pane in the GUI or in the **outflow** directory. Use GTKWave to view the waveform (see View Waveforms for instructions).

3.4 Build the Device Interface

Efinix Trion® FPGAs wrap a Quantum™-accelerated core (logic, memory, and multipliers) with a periphery that sends signals out to the device pins. The device periphery includes blocks such as GPIO pins, PLLs, and oscillators. You use the Efinity® Interface Designer to build the peripheral portion of your Trion® design.

In this section you use the Interface Designer to view the helloworld interface and add a missing GPIO resource.



1. Open the Interface Designer. The tool opens to a summary of your design and displays a list of interface blocks organized into categories.

   Note: The first time you launch the Interface Designer, it builds a cache and may be a little slow to load.

2. Review the assignments for `led[3]`.
   a) Expand **GPIO (6)**.

Tip: The number in parentheses indicates how many blocks your design implements in each category. For example the `helloworld` design has 6 GPIO pins.

   b) Select **led[3]**. The tool displays the resources led[3] uses in the Block Summary to the right.
   c) Click Show/Hide Editor to open the Block Editor if it is not already open.
   d) Click Show/Hide Resource Assigner to open the Resource Assigner table and review the settngs.
3. Generate constraints by clicking the Generate Efinity Constraint Files button.
   The software saves the interface design and creates the following output files in the outflow directory:
   - **helloworld.interface.csv—**Constrains the FPGA design pins used in the interface between the core and the periphery.
   - **helloworld.pt.rpt—**Provides information about the interface.
   - **helloworld.pinout.csv—**Contains the board design pinout in CSV format.
   - **helloworld.pt_timing.rpt—**Timing report for the Trion® interface logic.
   - **helloworld.pt.sdc—**Template SDC file to constrain the FPGA design pins based on the interface configuration.
   - **helloworld_template.v—**Template Verilog HDL file defining the FPGA design pins based on the interface configuration.
4. Close the Interface Designer and return to the Efinity main window.
5. Turn off the automated flow if it is not already off.
6. Click the Place dashboard button to run placement only. When placement completes, the Result pane displays under the dashboard.
7. Review the Results table, and notice that it reports your design has a missing interface pin.
8. Double-click **Result > Placement > helloworld.place.rpt**. The placement report opens.
9. Review the report. Under the IO Placement Summary, notice that `led[4]` is unassigned. It should be assigned to a GPIO output resource.
10. Open the Interface Designer.
11. Create a new block for `led[4]`.
    a) Select **Design : T8F81 > GPIO**.
    b) Click Create Block to create a new block instance.
    c) Enter `led[4]` as the instance name and press return.
    d) Choose output as the mode.
    e) Save.
    f) Click the Generate Efinity Constraint Files button. Before generating constraint files, the software performs a design check to look for problems in the design. In this example, you did not assign a resource to the new instance, so the software issues an error.
    g) Click **OK** in the message window. The Message Viewer opens and displays more detail about the design problem.
    h) Select **led[4]** in the Design Explorer.
    i) Open the Resource Assigner.
    j) In the Resource cell for led[4], type `GPIOL_21`. Alternatively, you can double-click in the Resource cell for led[4] and choose **GPIOL_21** from the resource list.
    k) Press Enter. The software assigns `led[4]` to GPIOL_21 and indicates the assignment in the resource list with a checkmark.
    l) Generate Efinity constraint files again.

12. Close the Interface Designer and return to the Efinity software.

3.5 Perform Place & Route

You perform place and route from the GUI or from the command line.

 Enable/disable automated flow  Place the design

1. In the GUI, turn the automated flow back on.
2. Click the Place dashboard button.

   If automated flow is off, the software goes through placement only; if it is on, the software goes through placement and routing and bitstream generation.

To perform place and route and generate at the command line, use the following command:

Linux:

```
> efx_run.py helloworld.xml --flow pnr
```

Windows:

```
> efx_run.bat helloworld.xml --flow pnr
```

View report files in the Results pane in the GUI or in the **outflow** directory.

4.0 Review Results

The software includes several tools with which you review and cross-probe results:

- Timing Browser
- Floorplan Editor
- Tcl Command Console

The software writes timing information to the **helloworld.timing.rpt** file in the **outflow** directory. You can also view this file in the GUI Results pane.

The helloworld SDC file defines the 33 MHz clock, `clk`, and a 5 ns input and output delay for all outputs relative to `clk`.

Note: To apply SDC changes to your design, you must recompile.

4.1 Review Place & Route Results in the Floorplan Editor

The Floorplan Editor provides a visual representation of the device. After you compile, the viewer shows the tiles used for logic, routing, memory, etc.


View Floorplan


World view


Show fanin


Show all nets


Net tracer


Show fanout


Clear net trace

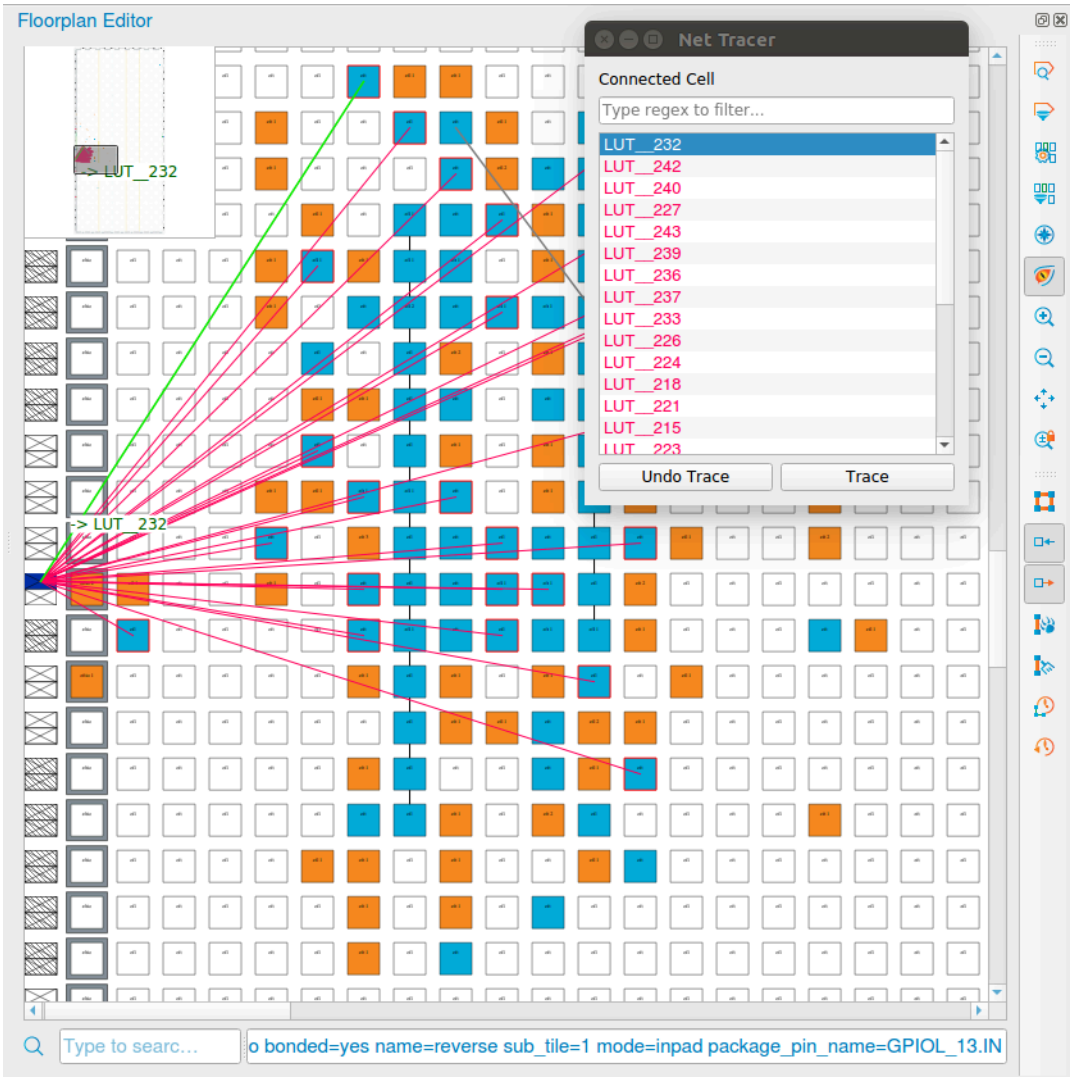

Toggle Floorplan Legend/Filter

Note: Close the Console or detach the Floorplan Editor for easier viewing.

1. Open the Floorplan Editor.
2. Turn on the World View to orient yourself in the floorplan.
3. Zoom in. The tiles with a solid color in the center are used blocks (blue for logic and orange for routing; toggle the Floorplan Legend/Filter to view the colors for other blocks). When you click a tile, the block coordinates and name are displayed.
4. Turn off Show All Nets.
5. With a logic tile selected, turn on Show Fanin and Show Fanout. The software shows fanout lines in red and fanin lines in blue.
6. Open the Net Tracer. The Net Tracer shows the cells that are connected to the selected cell. Double click a connected cell name to jump to that cell in the floorplan. The Net Tracer shows your path as you jump from cell to cell.
7. Close the Net Tracer. The traces persist when the Net Tracer window is closed.
8. Click Clear Net Trace to remove the tracing lines.
9. Click used logic tiles (blue) to observe the flow of logic through the tiles.

Note: Leave the Floorplan Editor open to explore timing in the next section.

Figure 2: Floorplan Editor



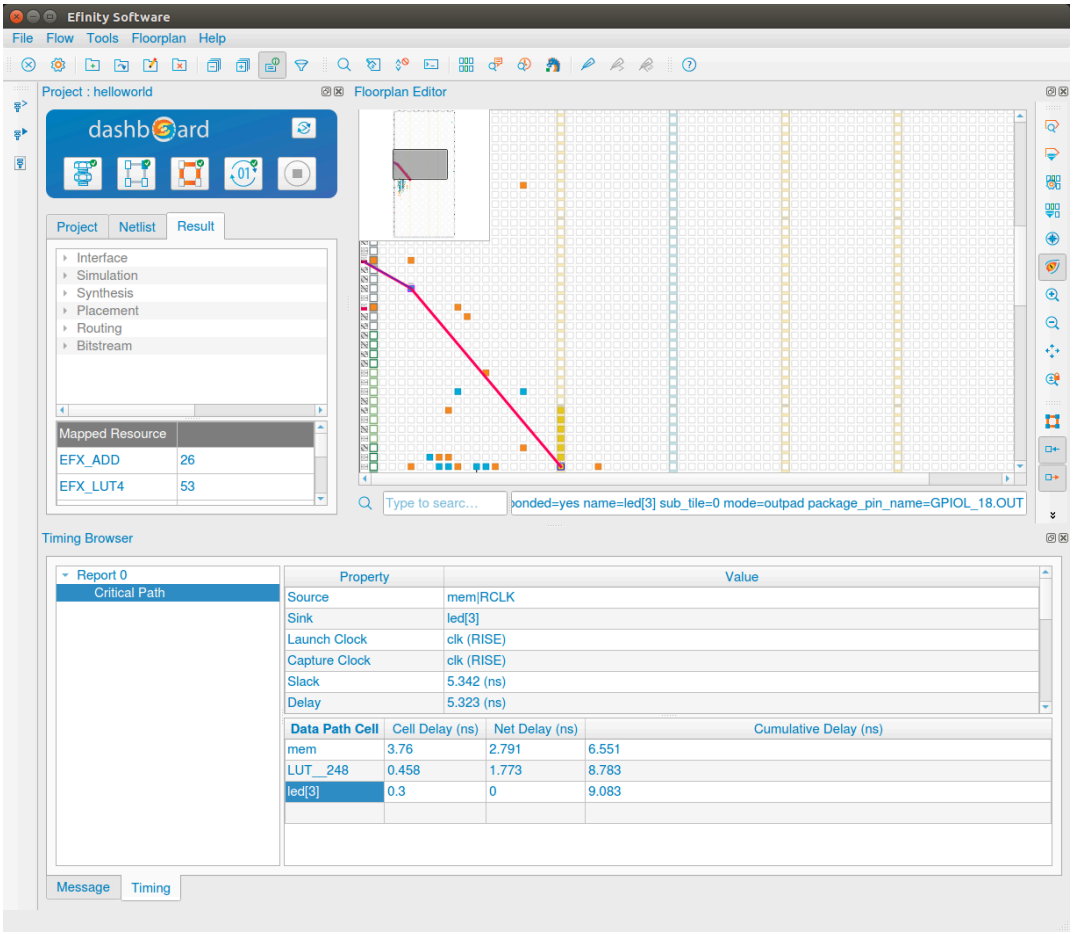
4.2 Use the Timing Browser

Use the Timing Browser with the Floorplan Editor to explore the your design's critical paths and the cells on those paths. When you first open the Timing Browser, it displays the most critical path (the path with the least slack) and the cells on that path.

 Timing Browser  Show timing paths  Show timing delay

1. Open the Floorplan Editor if it is not already open.
2. Open the Timing Browser.
3. Turn on Show Timing Paths.
4. In the Timing Browser, click `led[0]` under Data Path Cell. The Floorplan Editor shows the I/O pad that is the end of the critical path.
5. Turn on Show Timing Delay. The Floorplan Editor shows the delay for the cells and nets.

Figure 3: Exploring the Design's Timing



4.3 Use the Tcl Command Console

You use the Tcl Command Console to analyze and explore timing.

Tcl Command Console

1. Open the Console if you had closed it previously.
2. Open the Tcl Command Console.
3. Type `source example_report.tcl` to view the helloworld example timing reports in the console.

The Timing Browser displays the new reports. Click on paths in the browser to view them in the Floorplan.

Learn more: Refer to "Appendix B: Tcl Timing Report & Flow Commands" in the Efinity Software User Guide for a listing of available Tcl commands. For help on available Tcl commands, type `help -category <sdc or timing>` in the Tcl Command Console.

5.0 Configure the FPGA

In this section you install the USB driver for the programming cable and then configure the FPGA using the Efinity® Programmer.

Install the USB Driver

If you have not already done so, install the USB driver for the Efinix programming cable.

Linux: Use the command:

```
> sudo <installation directory>/bin/install_usb_driver.sh
```

Windows: Follow these instructions:

1. Download and install the Zadig software from zadig.akeo.ie.
2. Open the Zadig software.
3. Choose **Options > List All Devices**.
4. Turn off **Options > Ignore Hubs or Composite Parents**.
5. Select the development board you want to target.
6. Select **libusbK** (*version*) next to **Driver**. (Do *not* choose WinUSB)
7. Click **Replace Driver**.
8. Repeat step 2 for each unique JTAG device you want to target. For example, if you want to use both the T8 and T20 development boards, you must install 2 USB drivers, one for each board.

Efinix development boards have FTDI (FT2232) chips to communicate with the USB port. These chips have separate channels for SPI and JTAG. Therefore, you **must** install the driver for the composite parent, **not** for the individual SPI and JTAG interfaces. If you install the driver for each interface, each interface appears as a unique FTDI device, which will make it hard to select the correct port during programming. Hint: the interface names end with *(Interface N)*, where *N* is the channel number; do not choose these.

Note: To ensure that the USB driver is persistent across user sessions, run the Zadig software as administrator.

Configure Using the Programmer

You are now ready to configure the Trion FPGA on the Efinix Trion development board.

1. Connect the USB cable to the board and to your computer.
2. To configure on the command line, use the command:

   Linux:

   ```
   > efx_run.py helloworld.xml --flow program
   ```

   Windows:

   ```
   > efx_run.bat helloworld.xml --flow program
   ```
3. To configure using the Programmer GUI, launch the GUI by clicking the Programmer icon in the Efinity software or use the command line:
   - *Windows*—<*installation directory*>**\bin\efinity_pgm.bat**
   - *Linux*—Use the command `efinity_pgm.py`
4. Choose the T8 development board as the **USB Target**. The board name appears as **AVR USB HID DEMO**.
5. Click the Select Image File icon.
6. Browse to the **outflow** directory and choose <*helloworld*>**.hex**.
7. Choose **SPI Active** or **SPI Passive** configuration mode.
8. Click **Start Program**. The console displays programming messages.

When the software has finished programming:

1. Observe the 5 green LEDs sweep across the LED display.
2. Press SW3 to change the sweep direction.
3. Press SW2 (reset) to stop the LED movement. The LEDs resume sweeping when you release SW2.
4. When you are finished, disconnect the USB cable from the board and your computer.

Debugger Tutorials

The software includes a hardware Debugger to probe signals in your design via the JTAG interface. The Debugger has two perspectives: *Profile Editor* and *Debug*. The Profile Editor perspective is where you add debug cores manually. You can also view the settings of a Logic Analyzer core that you created with the Debug Wizard. The Debug perspective is where you perform debugging.

The Debugger includes two debug cores, Virtual I/O (`vio`) and a Logic Analyzer (`la`). You use a manual flow and the Profile Editor to configure Virtual I/O cores. You can use a manual flow or the Debug Wizard's automated flow to configure Logic Analyzer cores.

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

Note: The Debugger tutorials require the T20BGA256 development board and Efinity® software v2019.3 or higher.

Automated Debugging Flow

This tutorial walks you through the Debugger automated flow using an example **helloworld** design. This tutorial uses the 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. Rename the directory as **helloworld-dbg-auto**.
4. Connect the T20 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-auto** 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. 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**.
5. Click **Next**. The wizard generates a debug profile.

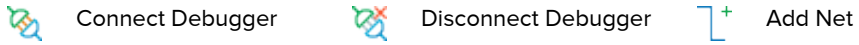
6. Leave **Enable "Auto Instantiation"** turned on. This option enables the debug profile in your project. Click **Finish**.
7. The software prompts you to recompile. Click **OK**.
8. Perform a full compile.

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>***.hex**.
5. Click **Start Programming**. The console displays programming messages.

Run the Debugger



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 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. Copy the folder *<Efinity path>*/**debugger/demo/helloworld-dbg** to your working directory.
3. Connect the T20 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.

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.
4. Leave the **Core name** as the default (**vio0**).
5. Add three probes (in) and three sources (out) with the following width settings (leave the other settings, including the probe and source names, at the defaults):

Name	Width
probe0	26
probe1	4
probe2	8
source0	1
source1	1
source1	4

6. Click **Add Debug Core > Logic Analyzer** to add a second core.
7. Leave the **Core name** as the default (**la0**).
8. Add three probes (in) with the same width settings as the probes in the VIO core (leave the other settings, including the probe names, at the defaults):

Name	Width
probe0	26
probe1	4
probe2	8

9. Click **Generate Debug RTL**. The Debugger creates the file **debug_top.v** and a template file in your project directory.
10. 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. Double-click the **helloworld.v** file in the Project tab to open it in the Code Editor.

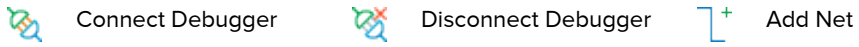
6. Edit the design to enable the debug code:
   a) Uncomment lines 18 - 28 (these lines enable the JTAG USER TAP interface ports).
   b) Uncomment lines 47 - 49 for the output probe signals.
   c) Comment out line 76.
   d) Uncomment line 77.
   e) Comment line 84.
   f) Uncomment line 85, which enables an output probe to act as a pushbutton.
   g) Uncomment lines 98 - 121, which instantiate the `debug_top` module.
   h) Save.
7. 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 constraints.
   f) Close the Interface Designer.
8. 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*>**.hex**.
5. Click **Start Programming**. The console displays programming messages.

Run the Debugger



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 Net.
3. Choose **probe2[3:0]** and click **OK**.
4. Specify a value of `0011`, which triggers when the LED output is 0011.
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 **source0** to **1** and press Enter. To stop the LEDs blinking, change the **Value** for **source1** to **1** and press Enter.
6. Click Disconnect Debugger to stop capturing data.

Revision History

Table 1: Revision History

Date	Version	Description
December 2019	5.0	Updated for the v2019.3 software. Added tutorials for the Debugger automated and manual flows. Added information on targeting ModelSim and NCSim for simulation.
August 2019	4.5	Updated for the v2019.2 software. Added command-line instructions for using the **efx_run.bat** command.
April 2019	4.4	Updated for the v2019.1 software.
January 2019	4.3	Updated for the v2018.4 software. Updated information on using the Programmer. Fixed typos.
October 2018	4.2	Updated for the 2018.3 software. Updated the Interface Designer steps. Added Python as an optional requirement. Minor changes throughout.
June 2018	4.1	Removed Python requirement; as of this release, Python is included with the software. Added the requirement that Windows users install the Microsoft Visual C++ 2015 x64 runtime library.
April 2018	4.0	Updated for v2018.0 software release. Added steps for using the Efinity Interface Designer. Added steps for programming with the Programmer GUI and Trion development board.
November 2017	3.1	Updated to target the Quantum family.
May 2017	3.0	Updated for v2017.0 software release. Renamed Floorplan Viewer as Floorplan Editor.
May 2016	2.0	Updated for v2016.0 software release. Documented Timing Browser and Tcl Console.
July 2015	1.1	Updated Floorplan Viewer information
May 2015	1.0	Initial release.