



Efinity[®] Unified Design Tutorial

UG-EFN-UNIFIED-v2.0
November 2025
www.efinixinc.com



Contents

- Introduction..... 3**
- Unified helloworld Project Files..... 3**
- Development Board Support..... 4**
- Open the helloworld Project.....4**
- Use Another Board..... 5**
- Enable the Unified Flow.....5**
 - syn_peri_port.....6
- Synthesize the Design..... 7**
- Perform Place and Route..... 7**
- Auto-Assign Resources..... 8**
- Simulate..... 9**
- Where to Learn More..... 10**
- Revision History.....11**

Introduction

The Efinity[®] software is different from other FPGA software tools because it places emphasis on system-level design. Each design has two parts:

- The core RTL design, which is implemented in XLR cells, RAM blocks, and multipliers or DSP blocks.
- The interface design, which is implemented in interface blocks at the periphery of the chip; interface blocks connect to package pins.

These two parts are connected by a signal interface. Because the core design is independent from the interface design, you can easily change from one device to another or change your system design without changing your RTL. Instead you simply adjust the interface to meet your new requirements. Other FPGA software tools tightly integrate these two parts of the design, making it more difficult to make system-level changes.

There are several ways to implement your interface design:

- Use the Interface Designer GUI to add interface blocks and configure them.
- Use the Efinity Python API to write a script to generate an interface.
- Import GPIO and similar block data with an Interface Settings File (**.isf**).
- Use the unified flow to let the Efinity software generate most interface blocks for you from the data it interprets from your RTL design.



Note: The unified flow is available in the Efinity software v2024.2 and higher.

This user guide explains how to use the unified flow using an example project called `helloworld`.

Unified helloworld Project Files

The unified `helloworld` project includes these files:

- **helloworld.v**—Verilog HDL design file.
- **helloworld_fc_tb.v**—Testbench for simulating the "full chip" design, the RTL as well as the interfaces.
- **helloworld_tb.v**—Testbench file for simulating the RTL design only.
- **helloworld-unified.xml**—Project XML file.
- **.isf**—Interface settings files that define the design's resource assignments for Efinix development boards.

`<device>_kit_manual.isf` defines the interface blocks that are not discoverable in the RTL and their assigned resources.

`<device>_kit_pins.isf` contains the resource assignments for the interface blocks discovered in the RTL.



Important: Use the files in the `<Efinity path>/examples/helloworld-unified` directory for this tutorial. These files allow you to target different Efinix development boards. The design files in `<Efinity path>/project/helloworld-unified` target the T8 F81 development board only.

Development Board Support

The Efinity software v2025.2 and higher includes example design files that target multiple Efinix development boards. The Efinity project files target the Titanium Ti60 F225 Development Board by default, but you can easily use another board by:

- Changing the target family and FPGA in the Project Editor.
- Importing the interface design for another board with the provided Interface Settings File (.isf).

The .isf files are named according to which board they support. For example, **Ti60F225_kit.isf** is for the Ti60 F225 Development Board. Most boards are supported.

Open the helloworld Project

The Efinity software includes a unified version of the `helloworld` project. Reminder: use the files in the `<Efinity path>/examples/helloworld-unified` for this tutorial.

 Open Project

1. Copy the folder `<Efinity path>/examples/helloworld-unified` to your working directory.
2. Run the Efinity software.
3. Click the Open Project button.
4. Browse to `<working directory>/helloworld-unified`.
5. Select **helloworld.xml**.
6. Click **Open**.

In the **Project** tab, you see the following files:

- **Design—helloworld.v**
- **Constraint—helloworld.sdc** and **Ti60F225_kit_pins.isf**

Open the **helloworld.v** file by double-clicking the filename. The file opens in the Code Editor. The code has these definitions for the inputs and outputs:

```
output [DWIDTH-1:0] led,
(* syn_peri_port = 0 *) input clk,
input                rstn,
input                reverse
```

When you compile with the unified flow, the software creates GPIO for the inputs and outputs and assigns them to resources automatically. If you do not want the software to create GPIO for a signal, use the `syn_peri_port` synthesis option. When you compile the `helloworld-unified` project, the software creates GPIO for `led`, `rstn`, and `reverse` and does not create anything for `clk`. Instead you will create a PLL for `clk` manually with the Interface Designer. The `(* syn_peri_port = 0 *)` attribute tells the software that you will define the resource manually.

Use Another Board

Project Editor

It is easy to use another board (instead of the Ti60 F225 Development Board). You change the family and device, and then add the corresponding **.isf** to your project.

1. Change the family and device:
 - a) Choose **File > Edit Project**. The Project Editor dialog box opens.
 - b) Choose the device family in the **Family** drop-down list box.
 - c) Click **Select** next to **Device**. The **Device Selector** dialog box opens.
 - d) Choose the device on your board and click **OK**.
 - e) Click the **Design** tab.
 - f) Under the **Constraint** section, select the **Ti60F225_kit_pins.isf** file. This file has the GPIO assignments.
 - g) Click the Delete Constraint File button.
 - h) Click the Add Constraint File button.
 - i) In the **Open Constraint File** dialog box, select the *<device>_kit_pins.isf* for your board.
 - j) Click **Open**.
 - k) Click **OK**.
2. Import the interface design (specifically, the PLL configuration):
 - a. Open the Interface Designer.
 - b. Choose **File > Import Design**. The Import Design dialog box opens.
 - c. Choose **Interface Scripting File (.isf)** and click **Next**.
 - d. Click the file button to browse for the *<device>_kit_manual.isf*.
 - e. Choose the *<device>_kit_manual.isf* for your board.
 - f. Click **Finish**. The Interface Design loads the settings.
 - g. Save.

Enable the Unified Flow

Project Editor

You enable the unified flow in your project settings.

1. Choose **File > Edit Project**. The Project Editor dialog box opens.
2. Click the **Design** tab.
3. Observe that the **Enable Unified Netlist Flow** option is turned on.
4. Click **OK**.

When you turn on the unified flow, the software enables the following synthesis options:

- **--peri-syn-inference**, which turns on the unified netlist inference flow.
- **--peri-syn-instantiation**, which turns on the unified netlist instantiation flow.

These options use the `syn_peri_port` synthesis attribute.



Note: If your design already has an interface file and you want to see the unified netlist, disable **--peri-syn-inference** in the Project Editor's **Synthesis** tab. Turning this option off tells the software not to infer inputs and outputs as interface blocks.

syn_peri_port

You apply the `syn_peri_port` attribute to top-module ports in your RTL. This option allows you control how inference is done on each I/O port. In the default inference flow, synthesis infers I/O registers or I/O buffers on any I/O port if it is possible. The tool prioritizes inferring I/O registers over I/O buffers (i.e., if an `EFX_IBUF` or `EFX_IREG` can be inferred for a given input port, an `EFX_IREG` will be inferred).



Learn more: Refer to the Trion[®], Topaz, and Titanium Quantum primitives user guides for primitive definitions.

The attribute has the following settings:

- 0—Do not infer any interface blocks
- 1—Infer buffers only (`EFX_IBUF`, `EFX_OBUF`, `EFX_IO_BUF`)
- 2—Infer registers only (`EFX_IREG`, `EFX_OREG`)
- 3—Infer buffers and registers (default value in inference flow)

This attribute is supported in the Efinity software v2024.2 and higher.

Verilog HDL:

```
module iregtst_port_attr (clk, i1, o1, i2, o2, i3, o3, i4, o4, i5, o5);
  (* syn_peri_port = 0 *) input clk;
  (* syn_peri_port = 0 *) input i1;
  (* syn_peri_port = 3 *) output o1;
  (* syn_peri_port = 2 *) input i2;
```

VHDL:

```
entity top is
  port (
    pll_resetn : out std_logic;
    pll_lock   : in  std_logic;
    clk        : in  std_logic;
    clk50      : in  std_logic;
  );
  ATTRIBUTE syn_peri_port : INTEGER;
  ATTRIBUTE syn_peri_port OF pll_resetn : SIGNAL IS 0;
  ATTRIBUTE syn_peri_port OF pll_lock   : SIGNAL IS 0;
  ATTRIBUTE syn_peri_port OF clk        : SIGNAL IS 0;
  ATTRIBUTE syn_peri_port OF clk50      : SIGNAL IS 0;
```

Synthesize the Design



Enable/disable automated flow



Synthesize the design

Run the synthesis step only:

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

You view the synthesis files in the **Synthesis** section of the **Results** pane in the GUI. These files are located in the project's **outflow** directory. In addition to the usual generated file, the unified flow creates:

- **helloworld.map.core.v**—The technology mapped netlist containing core logic that is passed to the place-and-route tool. The top-module name has `~core` appended to it. In this design, `helloworld` becomes `helloworld~core`.
- **helloworld.map.peri.v**—The technology mapped netlist containing interface logic instantiated in the RTL or inferred by synthesis. It also instantiates the core netlist. The top module is `helloworld`, and it instantiates `helloworld~core`.

Perform Place and Route



Enable/disable automated flow



Place the design

Run placement and routing.

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, routing, and bitstream generation.

Click the arrow to expand the **Interface** section of the **Results** pane in the GUI. These files are located in the project's **outflow** directory. In addition to the usual generated files, the unified flow creates:

- **helloworld.peri.rtl.v**—Netlist containing only the interface logic discovered by synthesis. The top module name is `helloworld`. This file contains the settings for the GPIO `clk`, `reverse`, and `led[5:0]`.
- **helloworld.peri.pt.v**—Netlist containing only the interface logic using the blocks created with the Interface Designer. The top module name is `helloworld~chip`. This file contains the settings for the PLL and `pll_refclk` GPIO block.

Because the project includes the `<device>_kit_pins.isf`, the software uses the `.isf` assignments to assign the GPIO to resources, except for the `clk` GPIO, which is specified in the Interface Designer.

Auto-Assign Resources

-  Project Editor
-  Delete SDC File
-  Interface Designer
-  Delete block
-  Place the design

In this step you delete the `<device>_kit_pins.isf` from your project and let the software assign resources automatically.

1. Open the Project Editor.
2. Click the Design tab.
3. Select the `<device>_kit_pins.isf` file in the Constraint table and click Delete Constraint File.
4. Click **OK**.
5. Open the Interface Designer.
6. Click the Clear Design button. The **Clear/Reset Design** dialog box opens.
7. Keep the default settings and click **OK**.
8. Click the Import Design button.
9. Choose **Interface Scripting File (.isf)** and click **Next**.
10. Click the Open File button.
11. In the **Import Design File** dialog box, choose the `<device>_kit_manual.isf` for your board. For example, choose `Ti60F225_kit_manual.isf` for the Ti60 F225 Development Board.
12. Click **Open**.
13. Click **Finish**.
14. Save the design.
15. In the Efinity main window, click the Place dashboard button to run the placement flow.

Click the arrow to expand the **Interface** section of the **Results** pane in the GUI. The unified flow creates an additional file:

- **helloworld.auto_asg.isf**—Contains auto-generated resource assignments for the interface instances.

Simulate

The unified flow supports simulation via the command line. By default, the Efinity software expects the following naming conventions:

- The testbench file. The software looks for a testbench file with the default name *<project name>_tb.v*; if you use the default name, you do not need to specify the testbench in the simulation command. In this design, the testbench file is named **helloworld_tb.v**. To use a different testbench name, use the `--tb` option to specify the filename.
- The testbench's top module. The software expects the top module to be named `sim`. To use a different top module name, use the `--tb_top` option.



Note: Run setup in the Efinity software directory before running simulation at the command line.

Tip: Add the testbench files to your project. Then you do not have to specify the testbench filename in the command.

RTL Simulation

Use this command to simulate the user RTL:

Windows:

```
efx_run.bat -f rtlsim helloworld.xml --un_flow
```

Linux:

```
efx_run.py -f rtlsim helloworld.xml --un_flow
```

Post-Map Simulation

Use this command to simulate the design after synthesis completes:

Windows:

```
efx_run.bat -f mapsim helloworld.xml --un_flow
```

Linux:

```
efx_run.py -f mapsim helloworld.xml --un_flow
```

Interface Simulation

Use this command to simulate the design after synthesis completes. This step is similar to post-map simulation, except interface register and buffer blocks are expanded into GPIOs.

Windows:

```
efx_run.bat -f ptsimrtl helloworld.xml --un_flow
```

Linux:

```
efx_run.py -f ptsimrtl helloworld.xml --un_flow
```

Full-Chip Simulation

Simulate the full-chip, including the RTL core, interface blocks discovered by synthesis, and interface blocks configured using the Interface Designer. The software expects the top module name for the testbench file to be `pt_sim`. You can specify a different top module name using the `--tb_top` option.

Note that the top module name for the full-chip Verilog HDL file **helloworld-unified.peri.pt.v** is `<top module>~chip`. In the helloworld-unified design, it is `helloworld~chip`. You need to modify your testbench file to instantiate `<top module>~chip` instead of `<top module>`.

Windows:

```
efx_run.bat -f ptsimfc helloworld.xml --tb helloworld_fc_tb.v --un_flow
```

Linux:

```
efx_run.py -f ptsimfc helloworld.xml --tb helloworld_fc_tb.v --un_flow
```

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 Software Installation User Guide](#)
- [Efinity Synthesis User Guide](#)
- [Efinity Timing Closure User Guide](#)
- [Efinity Trion Tutorial](#)
- [Efinity Debugger Tutorial](#)
- [Efinity Programmer User Guide](#)
- [Efinity IP Packager User Guide](#)
- [Trion Interfaces User Guide](#)
- [Titanium Interfaces User Guide](#)
- [Topaz Interfaces User Guide](#)
- [Efinity Interface Designer Python API](#)
- [Efinity Command-Line Interface User Guide](#)
- [Quantum® Trion Primitives User Guide](#)
- [Quantum® Titanium Primitives User Guide](#)
- [Quantum® Topaz Primitives User Guide](#)

In addition to documentation, Efinity 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
November 2025	2.0	Documents the new example design and flow that support multiple Efinity development kits. Added lines to Auto-Assign Resources on page 8 and Simulate on page 9. Updated icons and fixed typos. Added Where to Learn More on page 10.
May 2025	1.0	Initial release.