Efinity® Partitioning and Preservation Flows Tutorial

UG-EFN-TUTORIAL-PARTITIONING-v1.0
August 2026
www.efinixinc.com

Contents

Introduction

This tutorial guides you through three example flows to demonstrate use of the Efinity Partition Planner to support partitioning and preservation-based designs.

Partition Planner (Beta)

A design partition represents a portion of the RTL hierarchy located in a specific area in the device floorplan. Partitions ensure the quality and reproducibility of logic placement, routing, and timing results[1] across project and/or design iterations. Additionally, they enforce floorplanning requirements with placement constraints. Partitions support the following workflows:

- Team-based design (top-down methodology) where multiple users develop different parts of the design in parallel, reducing the overall development time.
- Design re-use (bottom-up preservation flow) in which you use partitions across projects and/or designs.
- Incremental compilation where you re-use compilation results thereby reducing compile times across project and/or design iterations.

The Efinity Partition Planner, which is beta in v2026.1, allows you to create and use partitions to support these workflows. As with a non-partition-based design, you should have a good idea of the design's interface and I/O requirements at the beginning of the design process. Ideally, you define partitions after determining the design interface.

Note: The Efinity software supports partitions for Titanium and Topaz FPGAs.

Tutorial Workflows

This tutorial walks you through several partition-based workflows using the Efinity Partition Planner. You use the included example projects to:

- Specify region constraints on parts of the design
- Perform incremental compilation and re-use parts of your design across projects
- Perform design core co-development in line with the team-based design (top-down) methodology

You can complete any or all of the example projects in any order.

This tutorial assumes that you have already installed the Efinity software v2026.1 or higher according to the instructions in the **Efinity Software Installation User Guide**.

Learn more: For more information about using the Efinity Partition Planner to create and modify partitions in new projects, see the **Efinity Software User Guide**.

[1] The software preserves timing results as long as you use the same timing constraints across projects/designs.

Place Region Constraints in the Efinity Software

The Efinity software v2026.1 or higher allows you to specify placement constraints on parts of your design. In this tutorial, you use the **r4000-place-regions** example project with the Partition Planner.

Open the Example Project

The **r4000-place-regions** example project is located in the *<Efinity installation path>*/**examples/r4000-place-regions** directory.

Note: By default, the Efinity software stores files in the user directory:

- *Linux*–**/home/***<user name>***/.efinity**
- *Windows*–**C:\Users***<user name>***\.efinity**

Browse to the software installation directory at *<Efinity installation path>***/examples** to find the example projects.

Open the Efinity GUI Open Project

1. Open the Efinity software.
2. Click the Open Project icon in the toolbar.
3. Browse to *<Efinity installation path>***/examples/r4000-place-regions**.
4. Select **r4000-place-regions.xml**.
5. Click **Open**.

Alternatively, use the command line to open the example project:

```
efinity --prj r4000-place-regions.xml
```

Open the Partition Planner

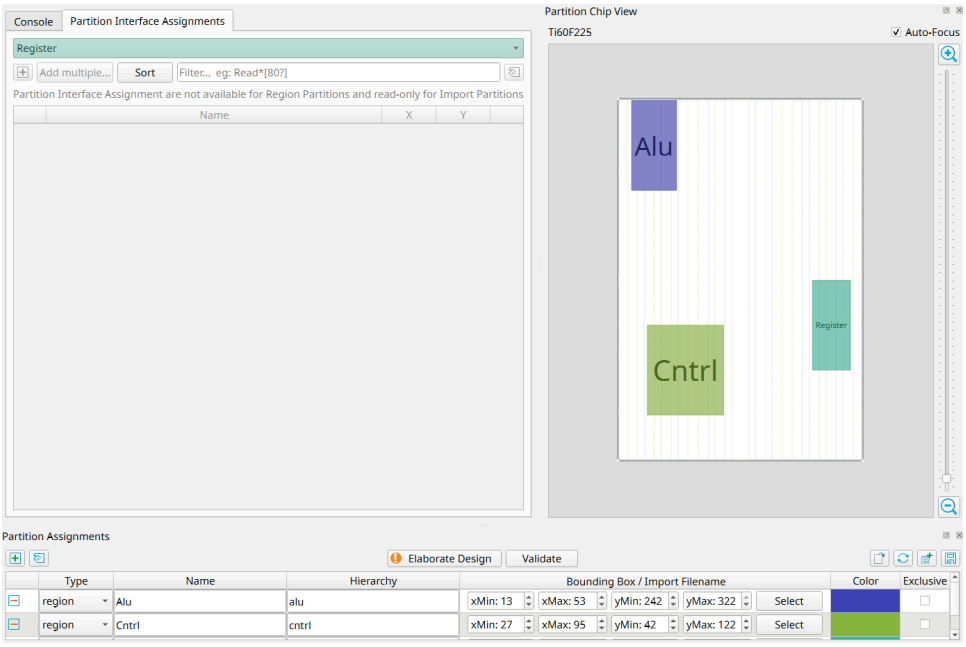
You specify placement constraints in the Partition Planner.


Partition Planner

Open the Partition Planner by clicking the toolbar icon or using the **Tools > Open Partition Planner** menu item.

The tool opens in the Efinity main window.

Figure 1: Partition Planner



Example Project Overview

The **r4000-place-regions** project contains a lightly modified version of the **r4000** example project in the *<Efinity installation path>*/**project/r4000** directory.

The **r4000** design contains a MIPS R4000 processor with three distinct modules: `mCntrl`, `mAlu`, and `mRegister`. Each module is instantiated once under the top-level module `mR4000` as `cntrl`, `alu`, and `register` respectively.

These instances have the corresponding design partitions `Cntrl`, `Alu`, and `Register`. The partition regions (bounding boxes) are spread farther apart in the example design for illustrative purposes.

Run the Flow

Enable/disable automated flow Synthesize the design

To compile the example project:

1. Turn on the automated flow.
2. Click the Synthesize icon in the Efinity Dashboard.

Alternatively, complete each stage of the software flow separately with the automated flow turned off.

Note: The Partition Planner window closes during compilation.

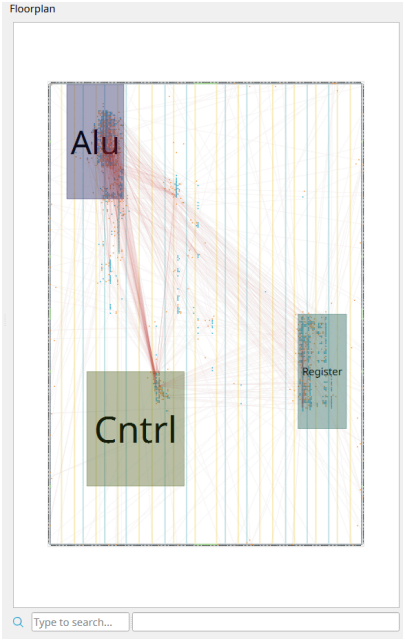
Open the Floorplan Viewer

You can observe the effects of placement constraints on the software flow by looking at the Floorplan Viewer, which is available after compilation.

View Floorplan

1. Open the Floorplan Viewer by clicking the toolbar icon or using the **Floorplan > View Floorplan** menu item.
2. Observe the partition region overlays for each of the design partitions:

Figure 2: Floorplan Viewer



Additionally, note that the cells for each partition are placed within the boundaries of the corresponding partition region.

Make Changes to the Example Project

To follow best practices, make a copy of the **r4000-place-regions** project in a separate work directory before you make any changes.

Adjust Partition Regions

Click and drag the boundaries of each partition on the **Partition Chip View** to interactively make adjustments.

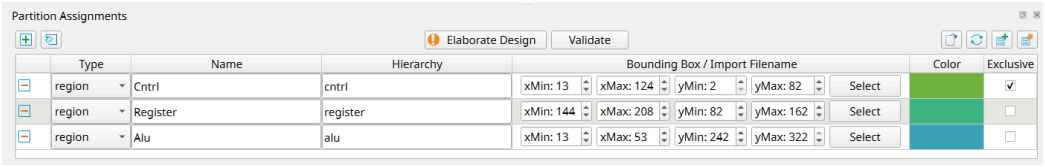
Learn more: See the Efinity Software User Guide for detailed information on specifiying and adjusting partition regions.

Toggle Partition Exclusivity

As visible in the Floorplan Viewer, the partition regions contain a subset of cells not belonging to any partition. This default behavior is intended to maximize device utilization and performance.

If you wish to toggle off this behavior, enable exclusivity for any design partition in the **Partition Assignments** table:

Figure 3: Toggle Partition Exclusivity



Validate and Save Changes to the Partition Configuration

Save

Click the Validate button above the **Partition Assignments** table to ensure the current partition configuation satisfies all design and device constraints.

If any failures occur, follow the instructions as printed on the Efinity Console, then perform the validation again. Once validation passes, click the Save icon to commit your changes.

Recompile the Design

Run the complete software flow from the Synthesis stage for any change to the design or partition configuration.

You do not need to recompile the design after changing the color of a partition. However, the updated partition color is not reflected on the partition overlay in the Floorplan Viewer until you perform a recompilation.

Run the Flow from the Command Line

As an alternative to the GUI software flow, command-line users can compile the example project on the command line by running the **RUN_DEMO** script included in the *<Efinity installation path>*/**examples/r4000-place-regions** directory.

```
cd <Efinity installation path/examples/r4000-place-regions
RUN_DEMO
```

Incremental Compilation and Core Re-use Flow in the Efinity Software

The Efinity software v2026.1 or higher allows you to perform incremental compilation and re-use parts of your design across projects. In this tutorial, you use the **r4000-mixed-preservation** example project with the Partition Planner.

Mixed Preservation Workflows

The incremental compilation workflow allows you to "lock down" the netlist, placement, and routing for parts of your design as you modify and iterate upon the remaining design. This process ensures the timing stability of the preserved portion of the design and additionally reduces compilation time.

The core re-use flow uses the same underlying technology as the incremental compilation workflow by migrating the preserved portion of the design to a new project. This allows you to re-use generic portions of your design for a new iteration of the project or for entirely different designs.

The generic partition export/import feature in the Efinity software v2026.1 enables both workflows.

Example Project Overview

The **r4000-mixed-preservation** example, located in the *<Efinity installation path>*/**examples/r4000-mixed-preservation** directory, consists of four Efinity projects named **step1** through **step4** and a **common** directory for shared source files.

```
r4000-mixed-preservation/
    common/
    step1/
    step2/
    step3/
    step4/
    RUN_DEMO
```

Note: By default, the Efinity software stores files in the user directory:

- *Linux*–**/home/***<user name>***/.efinity**
- *Windows*–**C:\Users***<user name>***.efinity**

Browse to the software installation directory at *<Efinity installation path>***/examples** to find the example projects.

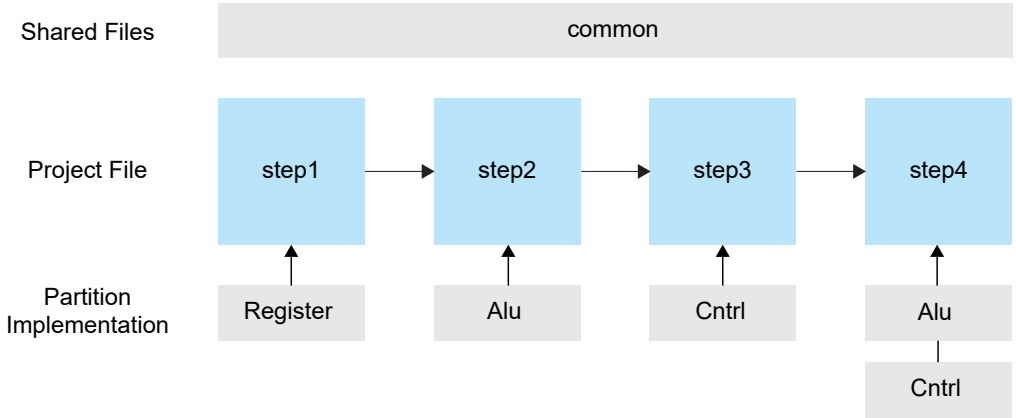
All projects contain a lightly modified version of the **r4000** example project in the *<Efinity installation path>***/project/r4000** directory.

The **r4000** design contains a MIPS R4000 processor with three distinct modules: `mCntrl`, `mAlu`, and `mRegister`. Each module is instantiated once under the top-level module `mR4000` as `cntrl`, `alu`, and `register` respectively.

These instances have the corresponding design partitions `Cntrl`, `Alu`, and `Register`.

The projects **step1** through **step4** represent a development process where design partitions are locked down as the **r4000** design is assembled, and where certain design partitions are re-used via export and import in new projects.

Figure 4: Mixed Preservation Development Flow



Project Step1

Begin by opening the **step1** project located in the *<Efinity installation path>*/**examples/r4000-mixed-preservation/step1** directory.

Open the Efinity GUI Open Project

1. Open the Efinity software.
2. Click the Open Project icon in the toolbar.
3. Browse to *<Efinity installation path>*/**examples/r4000-mixed-preservation/step1**.
4. Select **step1.xml**.
5. Click **Open**.

Alternatively, use the command line to open the example project:

```
efinity --prj step1.xml
```

In the first stage of this development process, the top level of the design is implemented in **r4000.v**. The source file **register.v** provides the implementation for partition `Register` (instance `register`).

The rest of the design (instances `ctrl` and `alu`) are provided as stubs and are unimplemented at this stage of development.

Figure 5: Project step1

Open the Partition Planner

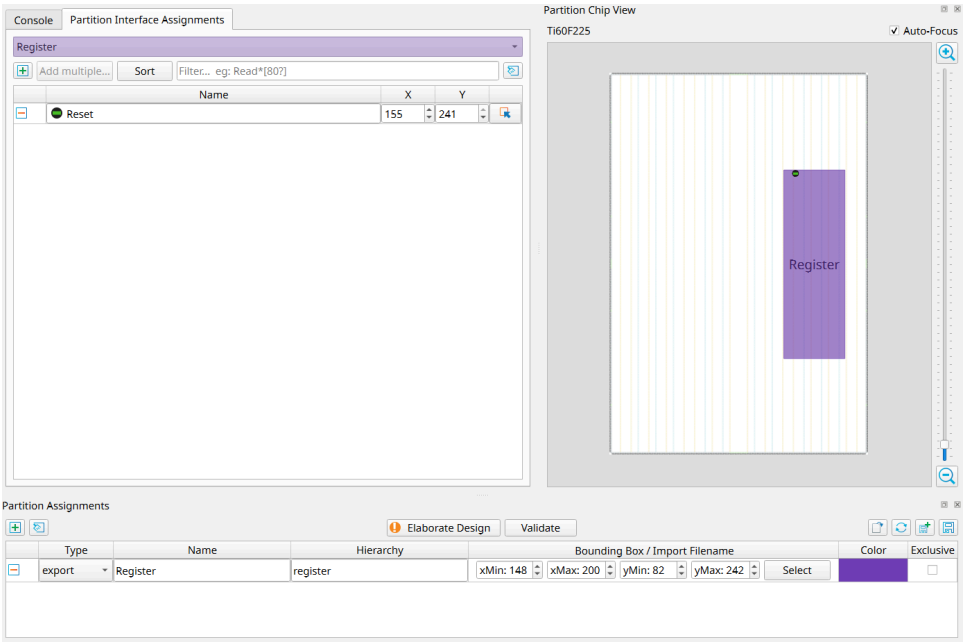
You specify export and import partitions in the Partition Planner.

Partition Planner

Open the Partition Planner by clicking the toolbar icon or using the **Tools > Open Partition Planner** menu item.

The tool opens in the Efinity main window.

Figure 6: Project Step1 Partition Planner



Observe that the `Reset` interface is explicitly placed within the `Register` partition.

Compile Step1

Enable/disable automated flow

Synthesize the design

To compile the example project:

1. Turn on the automated flow.
2. Click the Synthesize icon in the Efinity Dashboard.

Alternatively, complete each stage of the software flow separately with the automated flow turned off.

Note: The Partition Planner window closes during compilation.

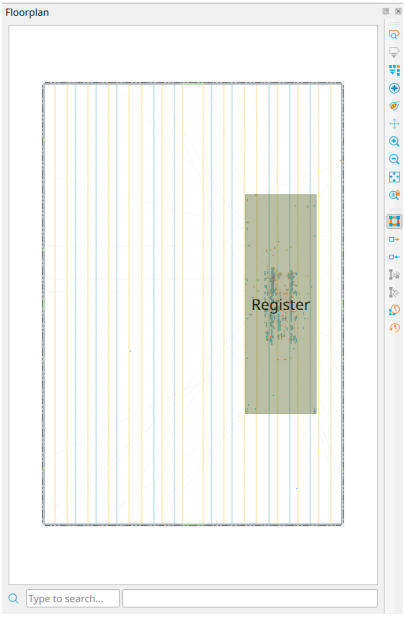
Open the Floorplan Viewer

You can observe the effects of export partition constraints on the software flow by looking at the Floorplan Viewer, which is available after compilation.

 View Floorplan

1. Open the Floorplan Viewer by clicking the toolbar icon or using the **Floorplan > View Floorplan** menu item.
2. Observe the partition region overlays for the `Register` partition:

Figure 7: Project Step1 Floorplan Viewer



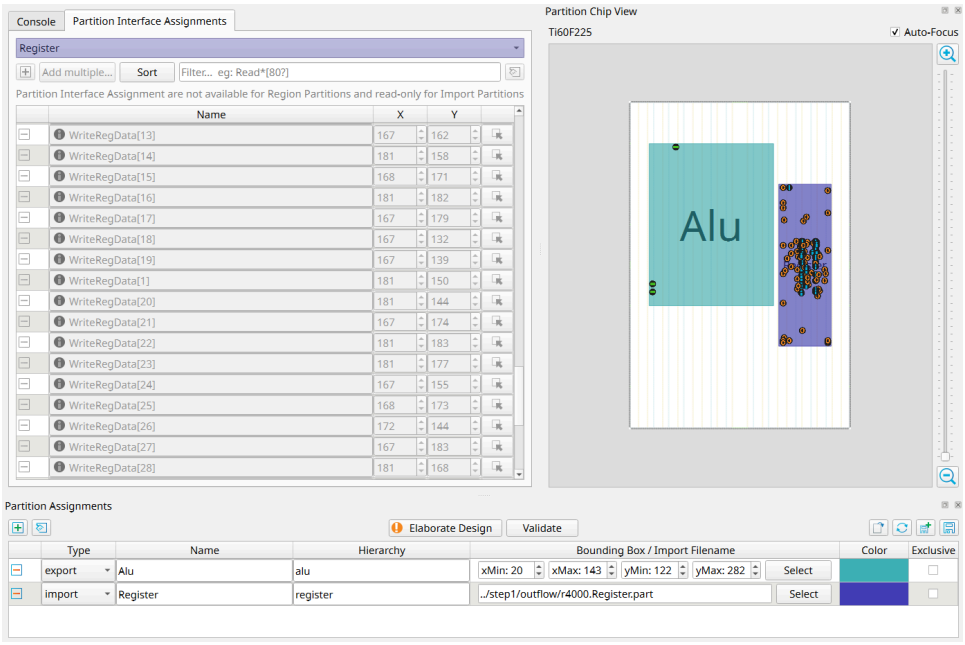
Observe that the cells for `Register` are placed within the boundaries of the corresponding partition region.

Project Step2

Continue by opening the **step2** project located in the *<Efinity installation path>*/**examples/r4000-mixed-preservation/step2** directory.

As in Project Step1 on page 10, open the Partition Planner:

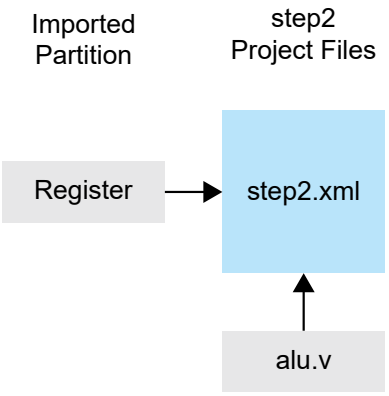
Figure 8: Project Step2 Partition Planner



During this stage of development, you re-use the `Register` partition from **step1** (see the specified import filename) while building the export partition `Alu` from the source files.

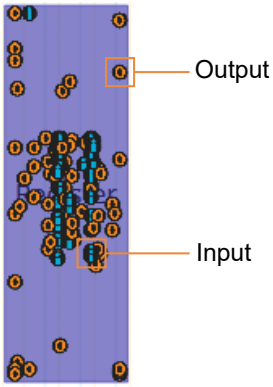
In this project, the the source file **alu.v** provides the implementation for the partition `Alu` (instance `alu`), while the partition `Register` is implemented as a stub (see **stubs.v** in the **common** directory). The contents of the stub `mRegister` module are replaced with the imported `Register` partition during place and route.

Figure 9: Project step2



In the **Partition Chip View**, observe the preview of the imported input (**i** icon) and output (**O** icon) interfaces on the partition `Register`. Three interfaces are also explicitly specified for the export partition `Alu`.

Figure 10: Partition Chip View Input/Output



Compile Project Step2 and Open Floorplan

Proceed to compile the **step2** project and open the Floorplan Viewer with the same method as as in Project Step1 on page 10.

Observe the imported cells from the `Register` partition:

Figure 11: Project Step2 Floorplan Viewer

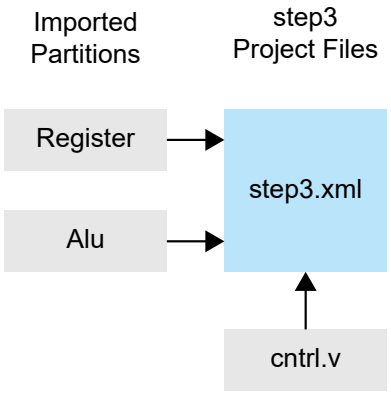
Recall that no source for `mRegister` is specified in this project.

Project Step3

In project **step3**, located in the *<Efinity installation path>*/**examples/r4000-mixed-preservation/step3** directory, you import the partition `Register` from **step1** as well as the partition `Alu` from **step2**.

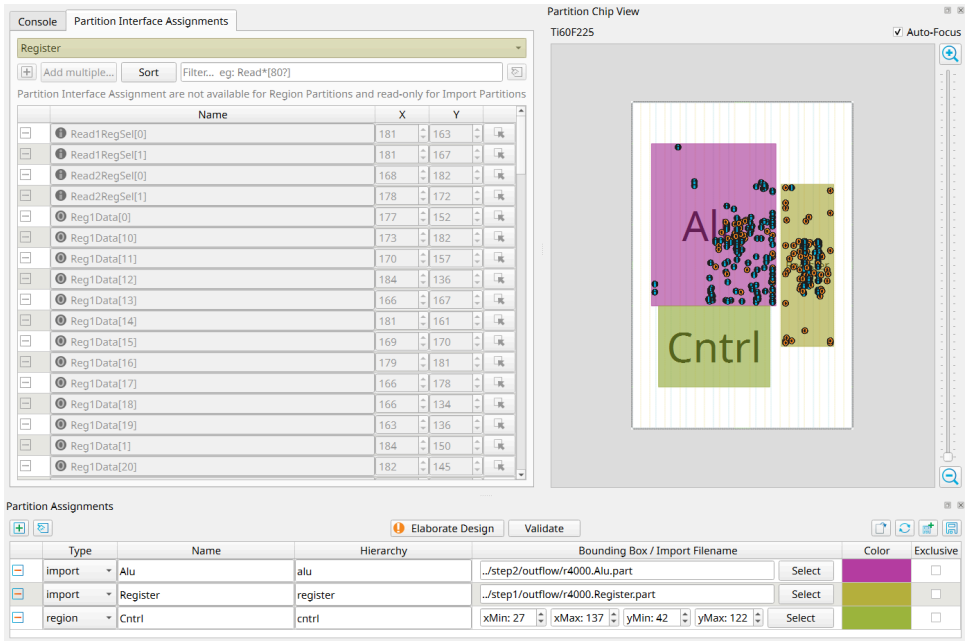
The source file **cntrl.v** provides the implementation for the partition `Cntrl` (instance `cntrl`). The partitions `Register` and `Alu` are implemented as stubs in this project.

Figure 12: Project step3



Open the Partition Planner:

Figure 13: Project Step3 Partition Planner



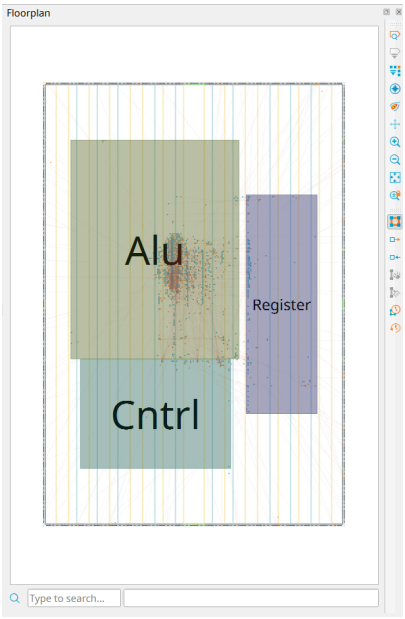
Note that the type (place) `region` is specified for the partition `Cntrl`, which means the placement and routing results for `Cntrl` are not exportable or importable. No partition archive file (**.part**) is produced for the `Cntrl` partition.

Compile Project Step3 and Open Floorplan

Proceed to compile the **step3** project and open the Floorplan Viewer.

Observe the imported cells from the `Alu` and `Register` partitions:

Figure 14: Project Step2 Floorplan Viewer



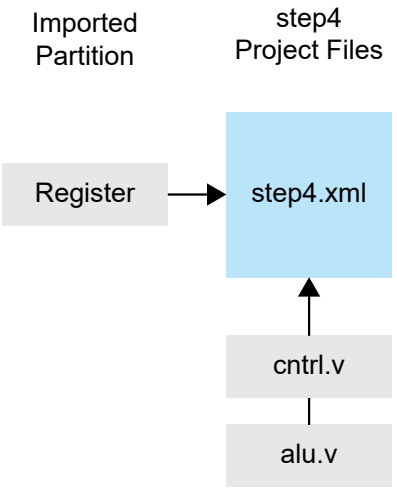
Recall that no sources for both partitions are specified in this project.

Project Step4

The project **step4**, located in the *<Efinity installation path>*/**examples/r4000-mixed-preservation/step4** directory, is a slight variation of project **step3** where the partition `Alu` is recompiled instead.

In this stage, you re-implement `Cntrl` and `Alu`. The source files **cntrl.v** and **alu.v** provide the implementations for the partitions `Cntrl` (instance `cntrl`) and `Alu` (instance `alu`) respectively. The partition `Register` is implemented as a stub.

Figure 15: Project step4



Open the Partition Planner, recompile, and observe the cells in the Floorplan Viewer as in the previous steps:

Figure 16: Project Step4 Partition Planner

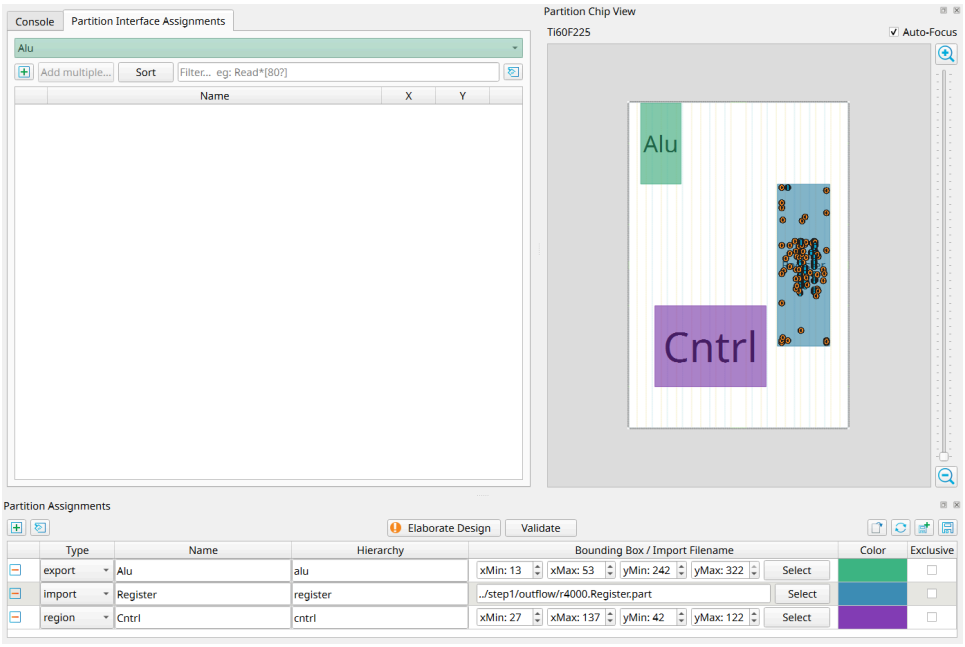
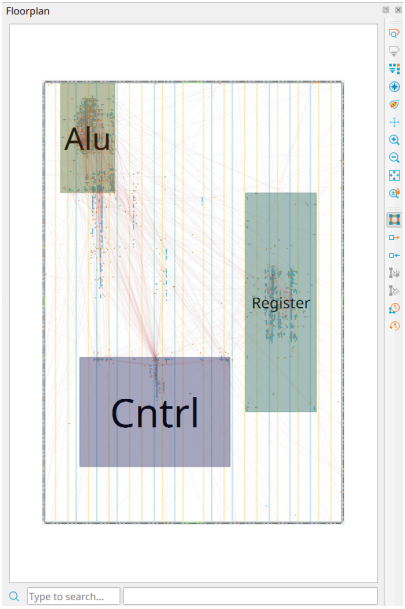


Figure 17: Project Step4 Floorplan Viewer



Practical Considerations for the Incremental Compilation Flow

In this example, you used a new project for every stage of development.

For the core re-use flow, it is considered good practice to maintain separate projects.

However, in the incremental compilation workflow, you may choose to stay on the same Efinity project. Flip the partition type from import to export as needed and specify the generated partition archive file (**.part**) in your output directory (usually **outflow**).

Run the Flow from the Command Line

As an alternative to the GUI software flow, command-line users can compile the example project on the command line by running the **RUN_DEMO** script included in the *<Efinity installation path>***/examples/r4000-mixed-preservation** directory.

```
cd <Efinity installation path>/examples/r4000-mixed-preservation
RUN_DEMO
```

Team-Based Design in the Efinity Software

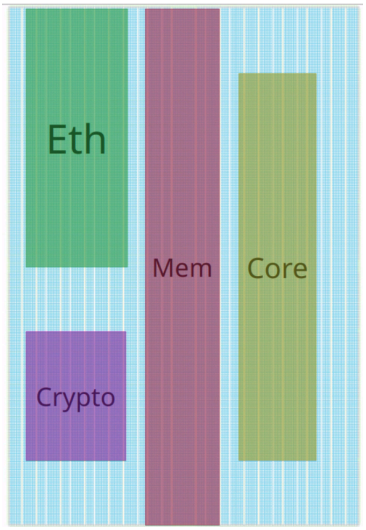
The Efinity software v2026.1 or higher allows you to perform design core co-development in line with the team-based design (top-down) methodology. In this tutorial, you use the **r4000-team-based-design** example project with the Partition Planner.

The Team-Based Design Methodology

The team-based design methodology, also known as the top-down or "waterfall" design methodology, is an EDA/CAD methodology that enables chip design collaboration and concurrent development among individual developers or larger teams.

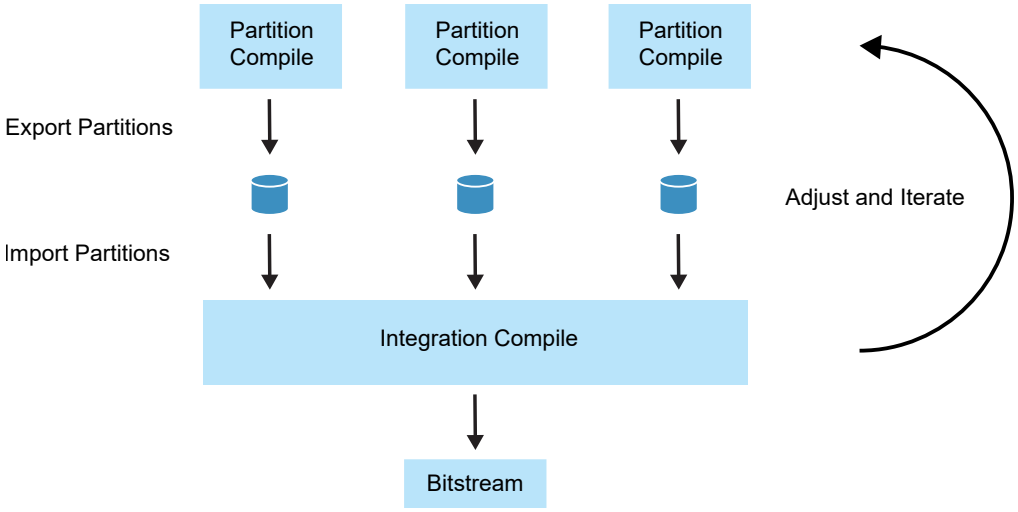
This is usually achieved by first dividing up the chip into separate partitions, each corresponding to a partition region, or bounding box. Partitions are usually defined to serve distinct design purposes (e.g., Ethernet block, crypto block, etc.).

Figure 18: Partition Regions



The design, validation, and timing closure of each separate partition can then be assigned to different developers or teams, who can work in parallel without the risk of overlapping device resources.

Figure 19: Team-Based Design Flow Chart



Note that *cores*, *IPs*, and *blocks* are also commonly used to refer to design partitions in the context of team-based design.

Once all the individual partitions are signed off, the integration team can complete the root partition, or design "top," connecting each partition's interface (outside-facing ports) to each other and to the boundary of the chip, as well as implementing any auxilary functions.

Motivations

The top-down approach is advantageous in that it allows for each design partition to be developed in parallel without the risk of resource conflicts, as the initial floorplanning step designates non-overlapping areas of the device for each partition. This facilitates true concurrent development and should lead to faster time to market.

This separation of functions usually improves design modularity, which can allow for core re-use in future projects.

Learn more: For more information about the core re-use flow, see the Efinity Software User Guide.

Preservation-based flows such as team-based design should also result in overall reductions in compilation time:

- The compilation workload is distributed, as the bulk of the compilation work is done by separate teams for their respective partitions in separate environments.
- Parts of the design are compiled separately rather than as a complete monolithic design, so the context of each compilation is smaller.
- During the integration compile, the cells and routes from imported partitions are preserved and are not considered for optimizations during placement and routing.

Example Project Overview

The **r4000-team-based-design** example, located in the *<Efinity installation path>*/**examples/r4000-team-based-design** directory, consists of five Efinity projects contained in the step directories and a **common** directory for shared source files.

```
r4000-team-based-design/
    common/
    step0-flrpln/
    step1a-alu/
    step1b-cntrl/
    step1c-register/
    step2-integ/
    RUN_DEMO
```

Note: By default, the Efinity software stores files in the user directory:

- *Linux*–**/home/***<user name>***/.efinity**
- *Windows*–**C:\Users***<user name>**\.efinity**

Browse to the software installation directory at *<Efinity installation path>***/examples** to find the example projects.

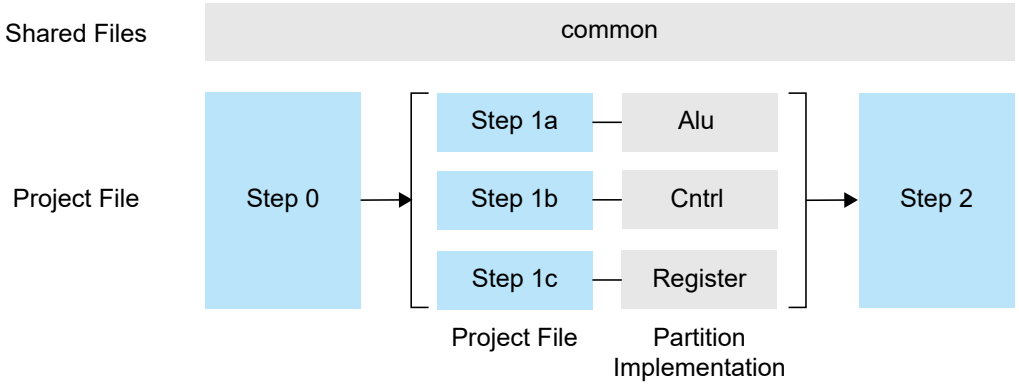
All projects contain a lightly modified version of the **r4000** example project in the *<Efinity installation path>***/project/r4000** directory.

The **r4000** design contains a MIPS R4000 processor with three distinct modules: `mCntrl`, `mAlu`, and `mRegister`. Each module is instantiated once under the top-level module `mR4000` as `cntrl`, `alu`, and `register` respectively.

These instances have the corresponding design partitions `Alu`, `Cntrl`, and `Register`. These are implemented and compiled in the **step1a**, **step1b**, and **step1c** projects.

step0-flrpln contains a skeleton project to facilitate the initial floorplanning step, while **step2-integ** is the final integration design.

Figure 20: Team-Based Development Flow



In the following sections, you run through each of these steps to demonstrate the team-based design methodology.

Step 0: Floorplanning

Begin by opening the **r4000_flrpln** project located in the *<Efinity installation path>*/**examples/r4000-team-based-design/step0-flrpln** directory.

 Open the Efinity GUI  Open Project

1. Open the Efinity software.
2. Click the Open Project icon in the toolbar.
3. Browse to *<Efinity installation path>*/**examples/r4000-team-based-design/step0-flrpln**.
4. Select **r4000_flrpln.xml**.
5. Click **Open**.

Alternatively, use the command line to open the example project:

```
efinity --prj r4000_flrpln.xml
```

The **r4000_flrpln** project contains a skeleton or early draft of the **r4000** design. While the design top module `mR4000` is fully implemented in this instance, the instances `ctrl`, `register`, and `alu` are provided as stubs and are unimplemented at this stage of development (see **stubs.v** in the **common** directory).

In step 0, you create design partitions around each of the aforementioned stub instances so they may be distributed for implementation in the following steps.

Open the Partition Planner

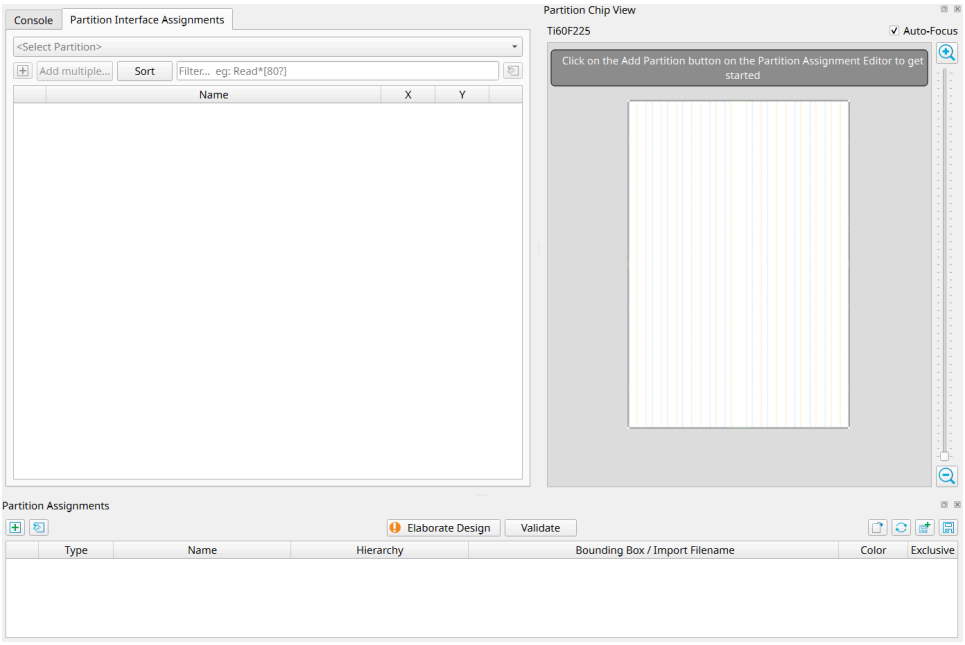
You specify export and import partitions in the Partition Planner.

 Partition Planner

Open the Partition Planner by clicking the toolbar icon or using the **Tools > Open Partition Planner** menu item.

The tool opens in the Efinity main window.

Figure 21: Partition Planner

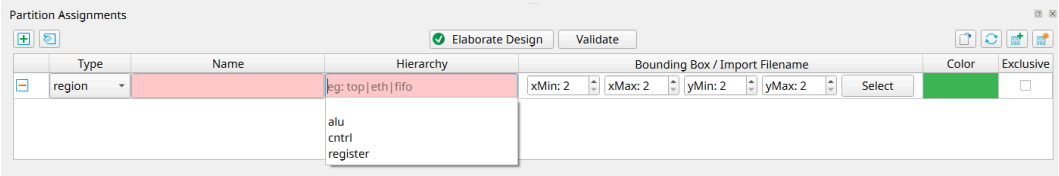


Create a Design Partition

Add Partition

1. Click the Add Partition icon.
2. Click the Elaborate Design button to get autocompletion hints for hierarchy paths.
3. Observe that the **Hierarchy** path cache has been populated.
4. In the **Partition Assignments** table, specify the partition name `Register` and hierarchy path `register`.
5. Use the **Type** dropdown to change the partition type to `Export`.
6. Click the Select button under the **Bounding Box / Import Filename** column.

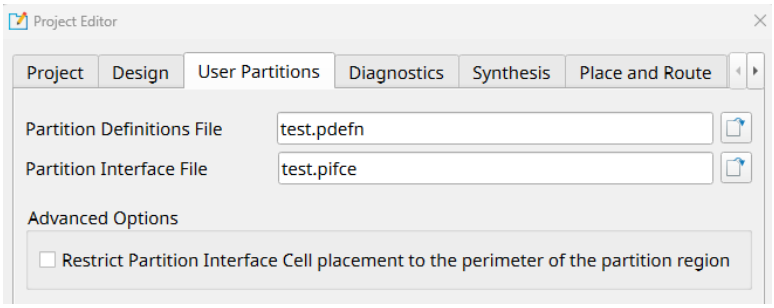
Figure 22: Partition Assignments Table



Save Load From

1. Return to the **Partition Chip View**.
2. Click, drag, and release to draw a partition region. Note that the draw operation always snaps to the closest valid box on the y-axis.
3. Click the Save icon.
4. Since no partition was previously defined, the software prompts you to specify filenames for the partition configuration files. Populate the **Partition Definitions File** and **Partition Interface File** fields.
5. Click the Save icon again.

Figure 23: Project Editor

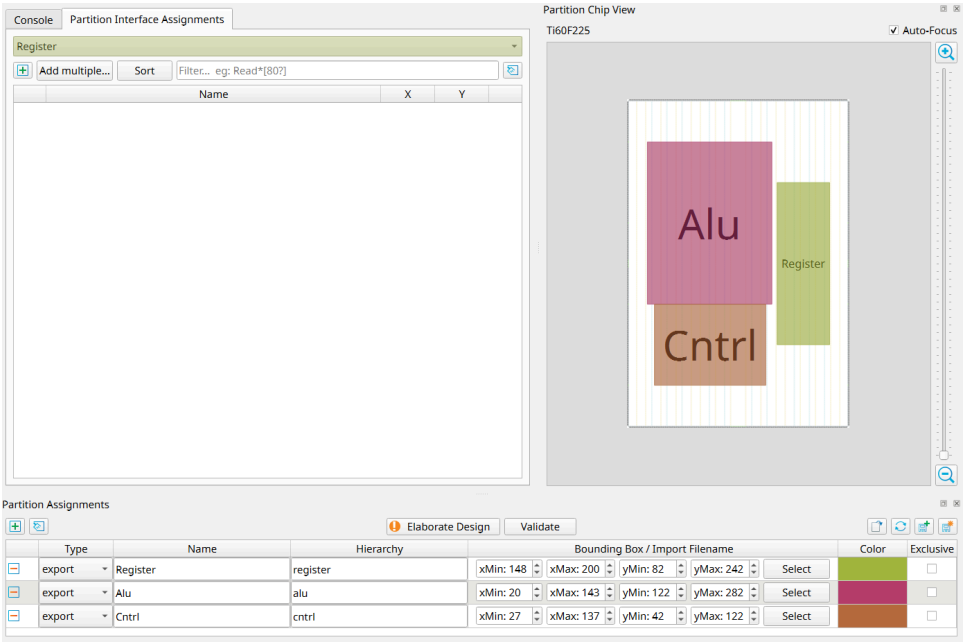


Open the partition configuration provided in the sample:

1. Click the Load From icon.
2. Select **example-configs/r4000.pdefn**.
3. The **Partition Assignment Editor** prompts you to specify a partition interface assignments file. Select **No**.

The partition assignments display in the **Partition Chip View**:

Figure 24: Step 0 Partition Planner



You use this partition configuration for the remaining steps.

Step 1: Partition Implementation and Compile

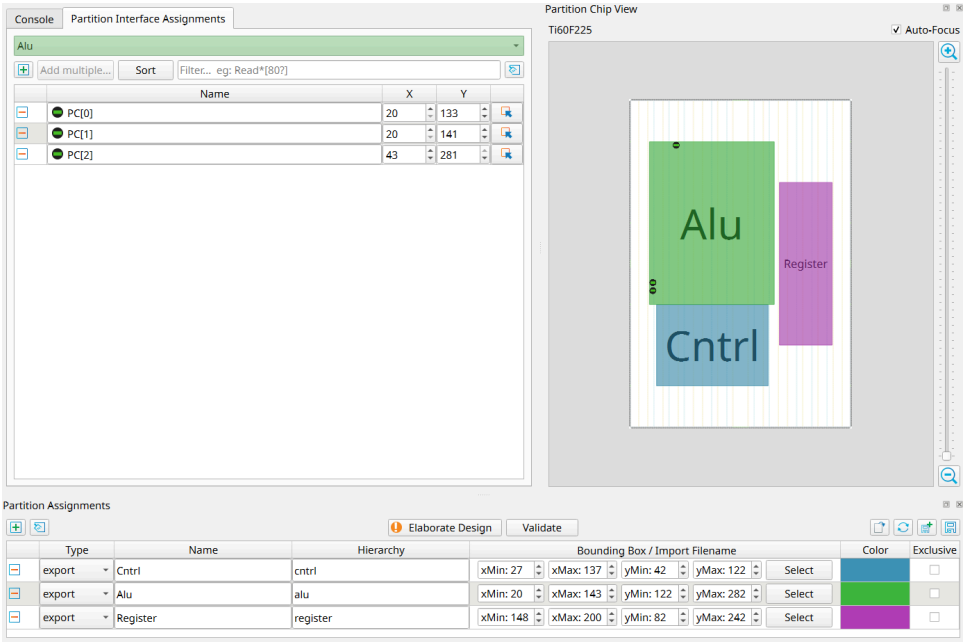
The three design partitions `Alu`, `Cntrl`, and `Register` can be developed in parallel, so you can complete the following sections in any order or even concurrently.

Step 1a: Partition Alu

Open the **r4000_alu** project located in the *<Efinity installation path>*/**examples/r4000-team-based-design/step1a-alu** directory.

Open the Partition Planner. Notice the project has manually specified the placement for a number of partition interfaces for `Alu`:

Figure 25: Step 1a Partition Planner



The source file **alu.v** provides the implementation for the partition `Alu` (instance `alu`). Partitions `Register` and `Cntrl` (instances `register` and `cntrl`) are implemented as stubs in this project.

Figure 26: Step 1a

Compile the Project

Enable/disable automated flow Synthesize the design

To compile the example project:

1. Turn on the automated flow.
2. Click the Synthesize icon in the Efinity Dashboard.

Alternatively, complete each stage of the software flow separately with the automated flow turned off.

Note: The Partition Planner window closes during compilation.

Open the Floorplan Viewer

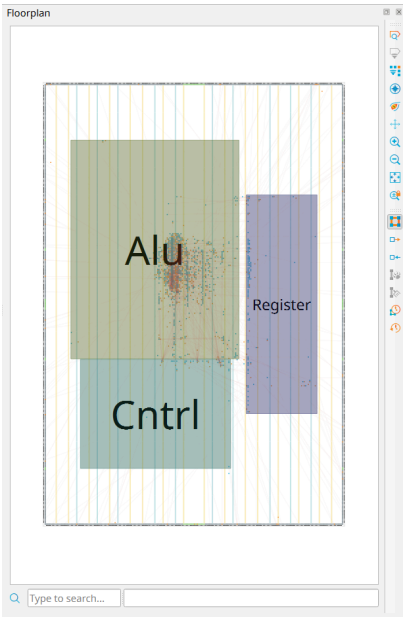
You can observe the effects of export partition constraints on the software flow by looking at the Floorplan Viewer, which is available after compilation.

View Floorplan

Open the Floorplan Viewer by clicking the toolbar icon or using the **Floorplan > View Floorplan** menu item.

See the partition region overlays for each of the partitions. Observe that the software places the cells for `Alu` within the boundaries of the corresponding partition region and that partitions `Register` and `Cntrl` are sparsely populated at this point:

Figure 27: Step 1a Floorplan Viewer

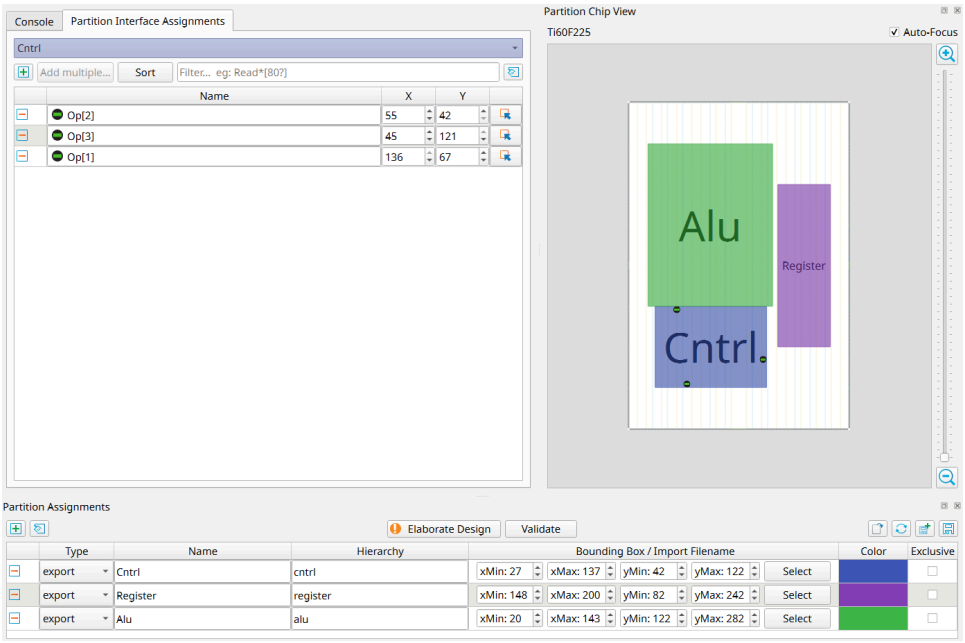


Step 1b: Partition Cntrl

Continue by opening the **r4000_cntrl** project located in the *<Efinity installation path>*/**examples/r4000-team-based-design/step1b-cntrl** directory.

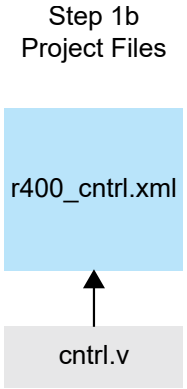
Open the Partition Planner. Notice the project has manually specified the placement for a number of partition interfaces for `Cntrl`:

Figure 28: Step 1b Partition Planner



The source file **cntrl.v** provides the implementation for the partition `Cntrl` (instance `cntrl`). Partitions `Register` and `Alu` (instances `register` and `alu`) are implemented as stubs in this project.

Figure 29: Step 1b

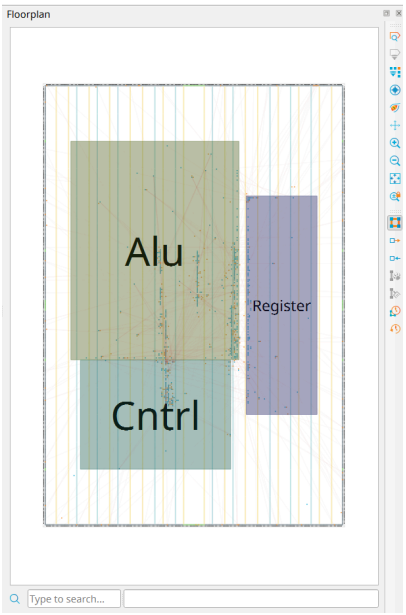


Compile the Project and Open Floorplan

Proceed to compile the project and open the Floorplan Viewer with the same method as as in **Step 1a: Partition Alu** on page 26.

Observe that the software places the cells for `Cntrl` within the boundaries of the corresponding partition region and that partitions `Register` and `Alu` are sparsely populated this time:

Figure 30: Step 1b Floorplan Viewer

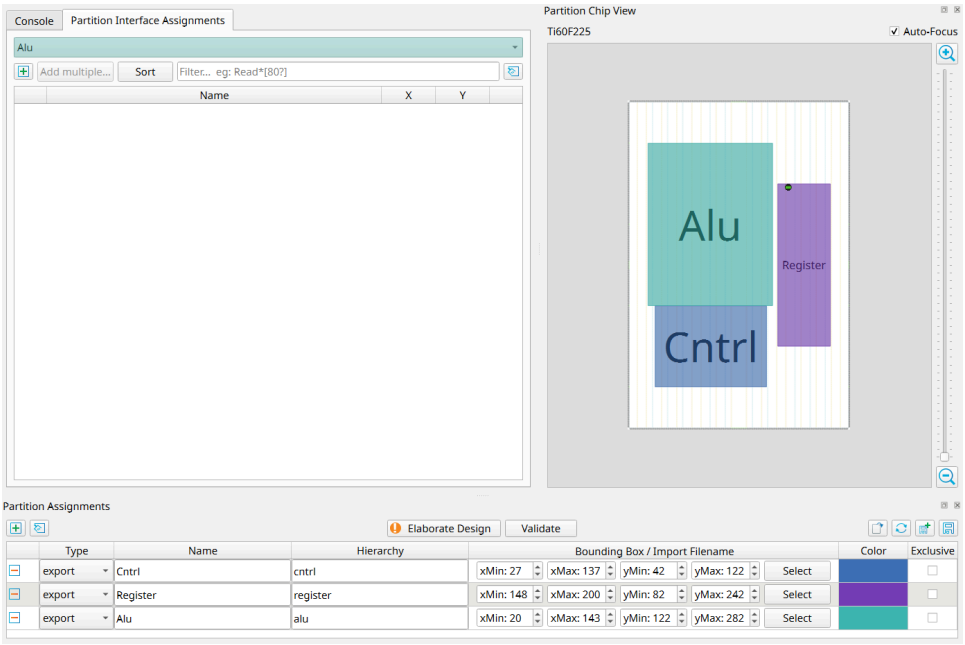


Step 1c: Partition Register

Finally, open the **r4000_register** project located in the *<Efinity installation path>*/**examples/r4000-team-based-design/step1c-register** directory.

Open the Partition Planner. Notice the project has manually specified the placement for a partition interface for `Register`:

Figure 31: Step 1c Partition Planner



The source file **register.v** provides the implementation for the partition `Register` (instance `register`) . Partitions `Cntrl` and `Alu` (instances `cntrl` and `alu`) are implemented as stubs in this project.

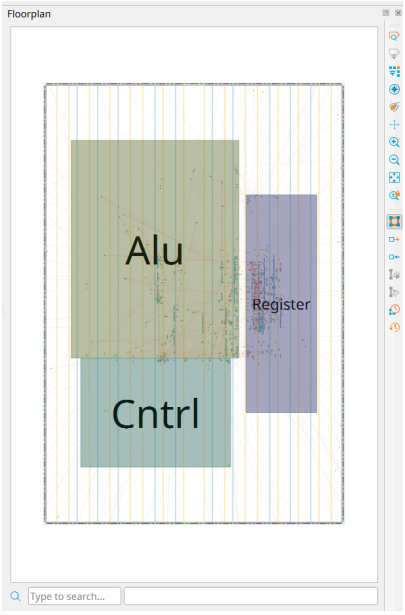
Figure 32: Step 1c

Compile the Project and Open Floorplan

Proceed to compile the project and open the Floorplan Viewer as in the previous steps.

Observe that the software places the cells for `Register` within the boundaries of the corresponding partition region and that partitions `Cntrl` and `Alu` are sparsely populated this time:

Figure 33: Step 1c Floorplan Viewer



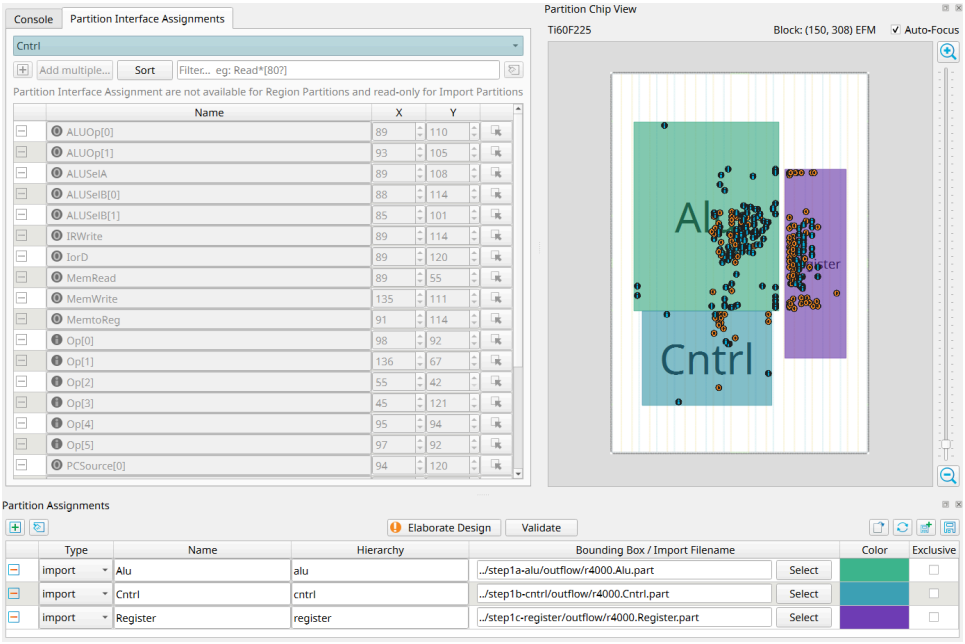
Step 2: Integration Compile

With all the design partitions implemented and compiled, you now bring them together in the final integration project.

Open the project **r4000_integ** located in the *<Efinity installation path>*/**examples/r4000-team-based-design/step2-integ** directory.

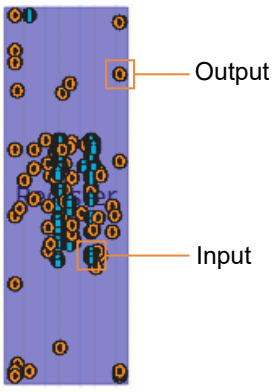
Open the Partition Planner:

Figure 34: Step 2 Partition Planner



This example project is prepared with all design partitions already changed to type `Import` and with listed filepaths to their respective partition archive files specified under the **Bounding Box / Import Filename** column. Also note the input (**i** icon) and output (**o** icon) interfaces on the **Partition Chip View**.

Figure 35: Partition Chip View Input/Output

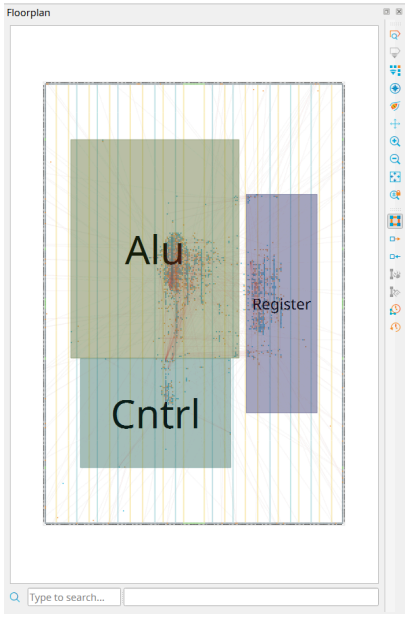


All design partitions are specified as stubs (unimplemented) in this project.

Compile the Project and Open Floorplan

Proceed to compile the project and open the Floorplan Viewer as in the previous steps.

Figure 36: Step 2 Floorplan Viewer



Observe that the software has imported the cells and routing from each partition archive. Recall that no implementations for any of the design partitions are available in this project.

Run the Flow from the Command Line

As an alternative to the GUI software flow, command-line users can compile the example project on the command line by running the **RUN_DEMO** script included in the *<Efinity installation path>*/**examples/r4000-team-based-design** directory.

```
cd <Efinity installation path>/examples/r4000-team-based-design
RUN_DEMO
```

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, 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
August 2026	1.0	Initial release.