SIEMENS EDA

A Quick Start to the Oasys-RTL™ Tool

Using an Example Design

Software Version 2022.2.R1

SIEMENS

About Siemens Digital Industries Software

Siemens Digital Industries Software is a leading global provider of product life cycle management (PLM) software and services with 7 million licensed seats and 71,000 customers worldwide. Headquartered in Plano, Texas, Siemens Digital Industries Software works collaboratively with companies to deliver open solutions that help them turn more ideas into successful products. For more information on Siemens Digital Industries Software products and services, visit www.siemens.com/plm.

Support Center: support.sw.siemens.com
Send Feedback on Documentation: support.sw.siemens.com/doc_feedback_form

Table of Contents

List of Figures

List of Tables

Chapter 1
Getting Started

A Quick Start to the Oasys-RTL Tool guides you through the Oasys-RTL design flow. Together with sample scripts and design data, it is a hands-on introduction to performing the basic functions of the Oasys-RTL tool.

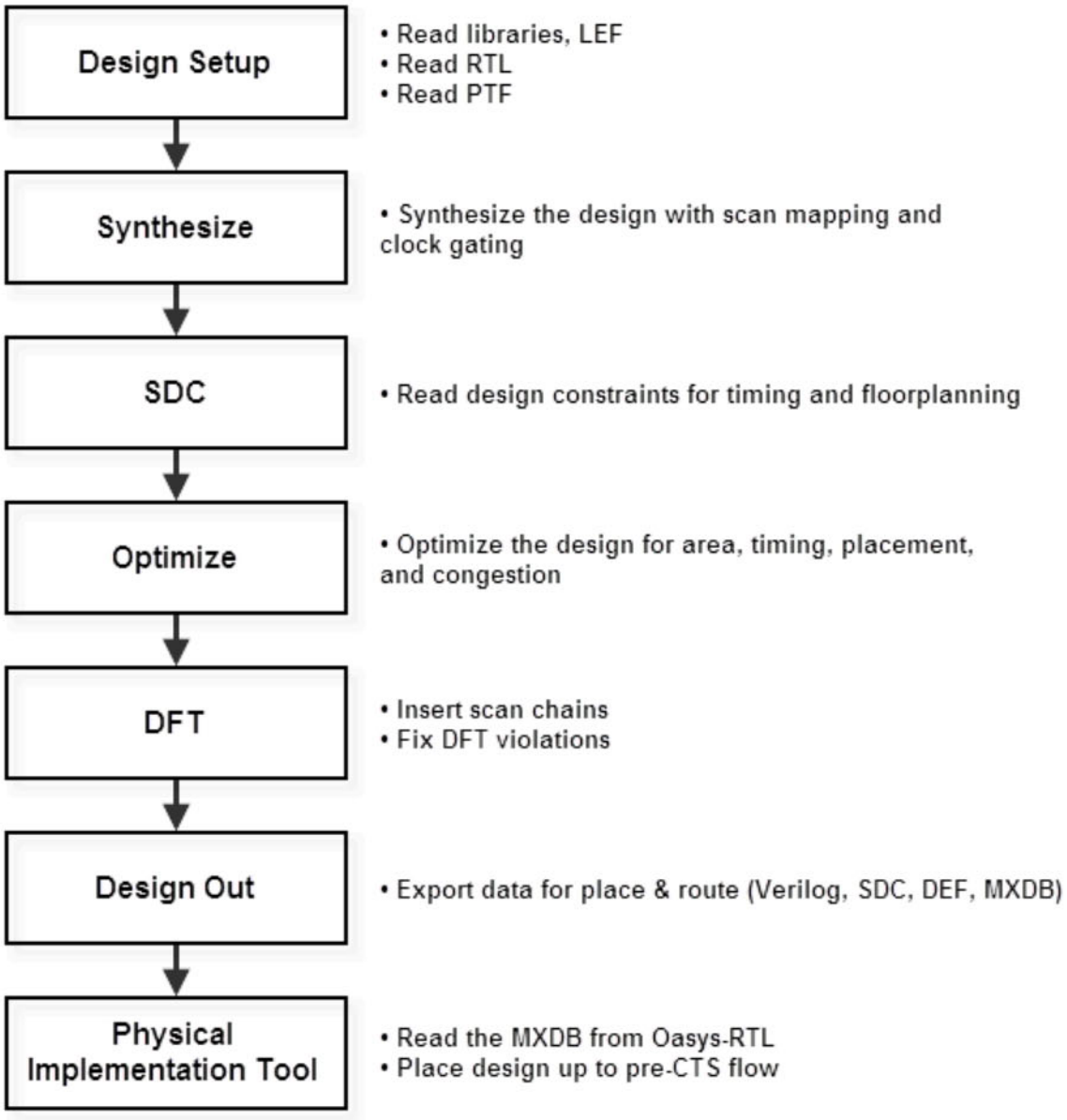
This module covers the following topics:

Oasys-RTL Flow

The Oasys-RTL tool provides an integrated flow for synthesizing a gate-level netlist from a behavioral RTL description.

Figure 1-1 shows the steps of this flow.

Figure 1-1. Oasys-RTL Physical Synthesis Flow



The chapters in this quick start guide show different aspects of this flow through examples.

Related Topics

Design Synthesis

Design Analysis

RTL Floorplanning

General Prerequisites

Ensure that your environment meets certain requirements before running the procedures in this document.

You need the following:

- Sufficient storage space for running the design.

Setting up the Quick Start Example Data

The Quick Start provides a companion demo design example that you can use to quickly learn the Oasys-RTL flow. All the steps in the Quick Start are based on this design data.

Prerequisites

- The *oasys_rtl_qs_ekit.tgz* file containing both this document and the design data. This file is located in the docs directory of the Oasys-RTL software tree. For example, *$OASYS_HOME/doc/oasys_rtl_qs_ekit.tgz*.
- Sufficient storage space for the design data and generated files. The recommended disk space is at least 800 MB for the design archive and generated files from the Oasys-RTL runs.

Procedure

1. Change to the directory from which you will run the examples.
2. Unpack the *oasys_rtl_qs_ekit.tgz* file:

   ```
   tar -xzvf oasys_rtl_qs_ekit.tgz
   ```

3. Change directory to *oasys_rtl_qs_ekit*:

   ```
   cd oasys_rtl_qs_ekit
   ```

   This directory contains the following:

   - *cleanup.sh* — Utility script for removing output files, if any, from previous runs.
   - *constraints* — SDC constraints file.
   - *demo_chip_rtl* — Contains the RTL files for the demo chip. This directory also includes the demo_chip.odb file which contains the optimized design from Chapter 2 (in case you can not download the OpenCore source files).
   - *docs* — Licensing information.
   - *libs* — Oasys library file (ODB) containing Liberty and LEF info for standard cells and macros/memories. Fastscan libraries for the Tessant DFT flow.

- *output* — Log, command, and other output files generated during an Oasys-RTL session.
- *scripts* — Scripts used in the quick start examples for Oasys-RTL operation.

4. Download the various modules of the demo_chip design from the "download" link on the following pages at the OpenCores website:

   - http://opencores.org/project,usb

     Name: usb

     Created: Sep 25, 2001

     Updated: Jul 27, 2013

   - http://opencores.org/project,nova

     Name: nova

     Created: Apr 21, 2008

     Updated: Jun 3, 2016

   - http://opencores.org/project,usb_phy

     Name: usb_phy

     Created: Sep 16, 2002

     Updated: Jan 11, 2017

   - http://opencores.org/project,hpdmc

     Name: hpdmc

     Created: Oct 26, 2008

     Updated: Aug 26, 2010

   - http://opencores.org/project,mips32r1

     Name: mips32r1

     Created: Oct 14, 2012

     Updated: Apr 12, 2014

5. Unpack the contents of all downloaded files into the following directory:

   ./demo_chip_rtl/rtl/

Note

Note the *rtl/* subdirectory above. If you unpack the design files into a different directory, Oasys-RTL may report errors when loading the design.

There are a few wrapper files included in the *oasys_rtl_qs_ekit.tgz* tar file under the following directories:

./demo_chip_rtl/rtl/demo_chip

./demo_chip_rtl/rtl/lib_cells

./demo_chip_rtl/rtl/mem_wrapper

Results

You have now installed the design data. Before starting the remaining chapters, review the items listed in General Prerequisites.

Chapter 2
Design Synthesis

The Oasys-RTL tool is used for synthesizing the design, which includes steps such as loading design inputs, synthesizing, optimizing, and exporting data.

Tip

If you want to skip the design setup procedure, you can bypass the first section.

Invoking the Oasys-RTL Tool and Preparing the Design for Synthesis

Load design data into the Oasys-RTL tool before running synthesis.

The synthesis and optimization scripts are located in the *scripts* directory. The *init_design.tcl* file sets up script variables and file paths and is called from within the *1_read_design.tcl* script. The *1_read_design.tcl script* reads the design data into the Oasys-RTL tool.

Prerequisites

- To enable Tessent integration with the e-kit, set the "dft_flow" variable to "Tessent" and set the "tessent_build" variable to the full path for the Tessent executable.

Procedure

1. Make sure that you are in the directory where you extracted the demo data. You should see the following sub-directories:
   - constraints
   - demo_chip_rtl
   - docs
   - libs
   - output
   - scripts

2. Enter the following command to invoke the Oasys-RTL tool:

   ```
   $OASYS_HOME/bin/oasys -log output/logs/synth.log
   ```

   You are presented with an Oasys-RTL shell:

   ```
   [oasys-RTL]$
   ```

3. For interactive execution, review and enter each command from the *scripts/1_read_design.tcl* file on the command line. Or for batch execution, source the entire Tcl file with the following command:

   source scripts/1_read_design.tcl

   This script file contains all of the commands to load the input design data into the tool.

   The tool issues this message when it completes:

   ```
   ------------------------------------
   Done preparing design for synthesis
   ------------------------------------
   ```

4. Keep the session open and proceed to the next step.

Results

The design files are loaded into the Oasys-RTL tool. The design optimization settings are set, and the tool is now ready to perform synthesis on the example design.

Related Topics

Synthesizing and Optimizing the Design

Synthesizing and Optimizing the Design

After you have loaded the design inputs and specified the relevant settings, synthesize and optimize the example design.

Procedure

1. If you are continuing with the previous session and the Oasys-RTL tool is still running, go to the next step. If you have not invoked the Oasys-RTL tool, enter the following command:

   ```
   $OASYS_HOME/bin/oasys -log output/logs/synth.log
   ```

2. The commands to synthesize the design are in the *scripts/2_synthesize_optimize.tcl* file. You can enter each command from the file or enter the entire Tcl file with the following command:

   source scripts/2_synthesize_optimize.tcl

   This step can take 15-25 minutes, depending on the characteristics of the machine used.

NOTE: If you did not download the RTL source from the Open Cores site, this script will load a previously optimized design (demo_chip_rtl/demo_chip.odb). This version of the design also contains scan chains. Using this ODB file will allow you to complete the steps in the tutorial but you will not be able to view or cross-probe to the RTL source.

3. While the tool is running, you can review the contents of the script *scripts/2_synthesize_optimize.tcl* and the output messages in the log file. Take note of the Oasys-RTL flow commands used to perform synthesis, and search on the items listed in Table 2-1.

Table 2-1. Search Strings to Review Progress of Synthesis, Optimization, and DFT

Search String	Description	Action
"config_shell -echo true"	Configures the shell to echo or log the commands. The config_shell command sets up different command shell preferences.	Confirm the setting.
"config_tolerance"	Controls the severity of RTL parser and elaboration errors allowed for the read_verilog, read_vhdl, and synthesize commands.	Check the messages after reading the RTL.
"synthesize -module demo_chip -map_to_scan -gate_clock"	Performs elaboration and mapping of the specified top module to the technology library. The -map_to_scan option directs synthesis to map the flops to scan flip-flops. If it is not specified, synthesize will not use scan flip-flops for registers. The -gate_clock option performs clock gating on enable flops.	Review the progress of synthesis (design mapping).
"report_design_metrics"	Display design related information.	Review the values for Total Instances, Blackboxes, Total Number of Buffers and Inverters, Area, and Utilization.

Table 2-1. Search Strings to Review Progress of Synthesis, Optimization, and DFT (cont.)

Search String	Description	Action
"optimize -virtual"	Performs virtual optimization that provides a first check of feasibility to close timing prior to full optimization in the physical context. Running a regular optimization after a virtual optimization is recommended only if timing is met in virtual placement.	If timing is not met after virtual optimization, identify and fix the bottlenecks in the RTL or constraints before re-running synthesis. Review the successive occurrences of optimization messages. Check messages for area, timing, and execution time information.
"optimize -place"	Starts placement optimization that optimizes based on floorplan and physical placement.	Check report_timing before "optimize -place" and review the successive occurrences of timing-driven placement messages.
"optimize"	Starts optimization that attempts to fix all timing violations in the physical context of the design. The result is a globally placed design that is optimized for timing, area, congestion, and power.	Review worst negative slack (WNS) for timing optimization. Also observe total congestion, slack, and average percentage of overflow.
"report_timing" "report_path_groups"	Reports timing and path groups after optimizations.	Notice slack before and after optimization for qualitative analysis. Review the same items after scan chain connections.

4. Close the *2_synthesize_optimize.tcl* file. When the run is complete, the following message is printed to the shell window:

```
-------------------------------------
Synthesis, optimization, DFT complete
-------------------------------------
```

5. If you want to insert scan chains into the design using Oasys or Tessent, you can source one of the following scripts: This step may take approximately 10 minutes.

```
source ./scripts/oasys_dft.tcl
source ./scripts/oasys_tessent_dft.tcl
```

6. Keep the session open and proceed to the next step.

Results

The flow script executed in this procedure synthesizes and optimizes the demo design. The tool saves the post-optimized design in the native Oasys-RTL database format (*.odb*) in the *<oasys_rtl_qs_ekit>/output/odb* directory.

Related Topics

Exporting the Design

Exporting the Design

Export your synthesized design to different formats used by other tools.

Procedure

1. The commands to export the design are in the *scripts/3_export_design.tcl* file. You can enter each command from the file or enter the entire Tcl file with the following command:

   source scripts/3_export_design.tcl

   This script writes out the synthesized Verilog, ODB, DEF, SDC, and DFT files from the Oasys-RTL tool.

   The script echoes the following message to the shell when it completes:

   ```
   -------------------------------------
   Design data exported to output dir
   -------------------------------------
   ```

2. At the command line prompt, type:

   exit

3. Proceed to Chapter 3, "Design Analysis".

Results

The design is optimized. Output files for place-and-route hand-off are generated and saved under the *<oasys_rtl_qs_ekit>/output/* directory as follows:

- **Synthesized Verilog** — *demo_chip.oasys_final.v*
- **SDC** — *demo_chip.oasys_final.sdc*
- **DEF** — *demo_chip.def*
- **ODB** — *odb/demo_chip.oasys_final.odb*
- **CTL** — *demo_chip.ctl*
- **STIL** — *demo_chip.stil*

If you downloaded the RTL source from the Open Cores site, the script writes a design ODB file at the end of each major RTL synthesis stages to the *<oasys_rtl_qs_ekit>/output/odb* directory as follows:

- **Post-synthesis** — *demo_chip.syn.odb*
- **Post-Virtual Optimized**— *demo_chip.virtual_opt.odb*
- **Post-Placement Optimized**— *demo_chip.placed_opt.odb*

Related Topics

Synthesizing and Optimizing the Design

Chapter 3
Design Analysis

The Oasys-RTL graphical user interface can be used to view and analyze a synthesized design.

Mouse Operations

The mouse operations available in the Oasys-RTL GUI change depending on the current view.

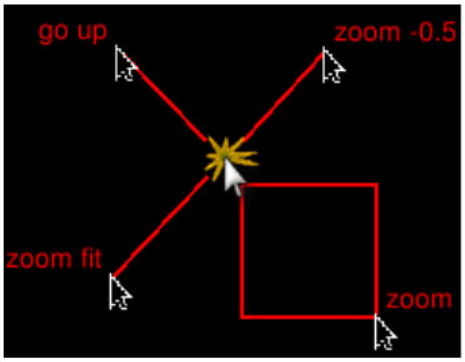
In the Physical view and other tabs, the mouse buttons function as follows:

- **Left-click —** Selects an object; a click-and-drag zooms to the rectangular area drawn.
- **Right-click —** Opens a context-sensitive options menu.
- **Keyboard Shortcuts —** You can also use the keyboard shortcuts listed in Table 3-1.

Table 3-1. Keyboard Shortcuts for Navigating

Key press	Action
Ctrl-plus (Ctrl-Shift-=)	Zoom in
Ctrl-hyphen	Zoom out
Ctrl-0	Zoom fit
Ctrl-1	Zoom selected

In the Netlist view, clicking and dragging the left mouse button gives you different navigation options, depending on the direction that you drag the mouse:



- **go up** — Pops the netlist view to the parent of the current hierarchy.
- **zoom -0.5** — Zooms out by half of the current area.
- **zoom** — Selects an area to enlarge to fit the current window.
- **zoom fit** — Fits the current schematic into the window.

Using the Oasys-RTL GUI

Launch the GUI, load a previously generated Oasys-RTL database (*<oasys_rtl_qs_ekit>/output/odb/demo_chip.oasys_final.odb*), and navigate the design.

Procedure

1. Enter the following command to invoke the Oasys-RTL GUI:

   ```
   $OASYS_HOME/bin/oasys -gui -log output/logs/analysis.log
   ```

   Tip

   Use the start_gui command if you are already in an Oasys-RTL shell.

2. Open the synthesized and optimized Oasys-RTL database that you generated in Chapter 2, "Design Synthesis", by sourcing the following script at the Oasys-RTL command prompt:

   source scripts/open_database.tcl

   The current status of the loading process is reported to the Oasys-RTL shell view. At the end of the script, Oasys reports the worst slack paths in the design.

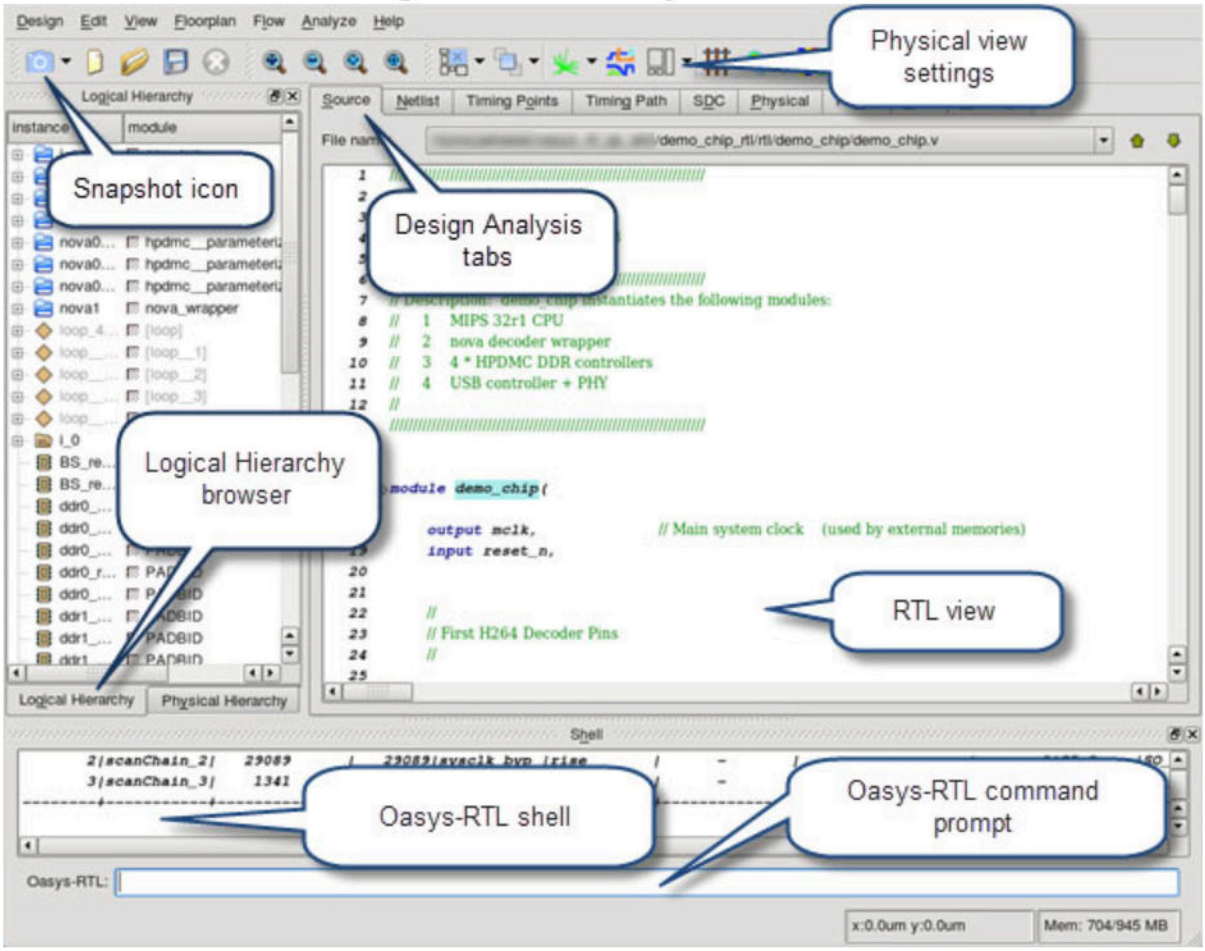
   When the database is completely loaded, the source netlist view of the top module is in the center of the main window. Instances that belong to the top module are in the left pane.

Note

If you did not download the RTL source from the Open Cores site, the *open_database.tcl* script will load a previously optimized design (demo_chip_rtl/demo_chip.odb).

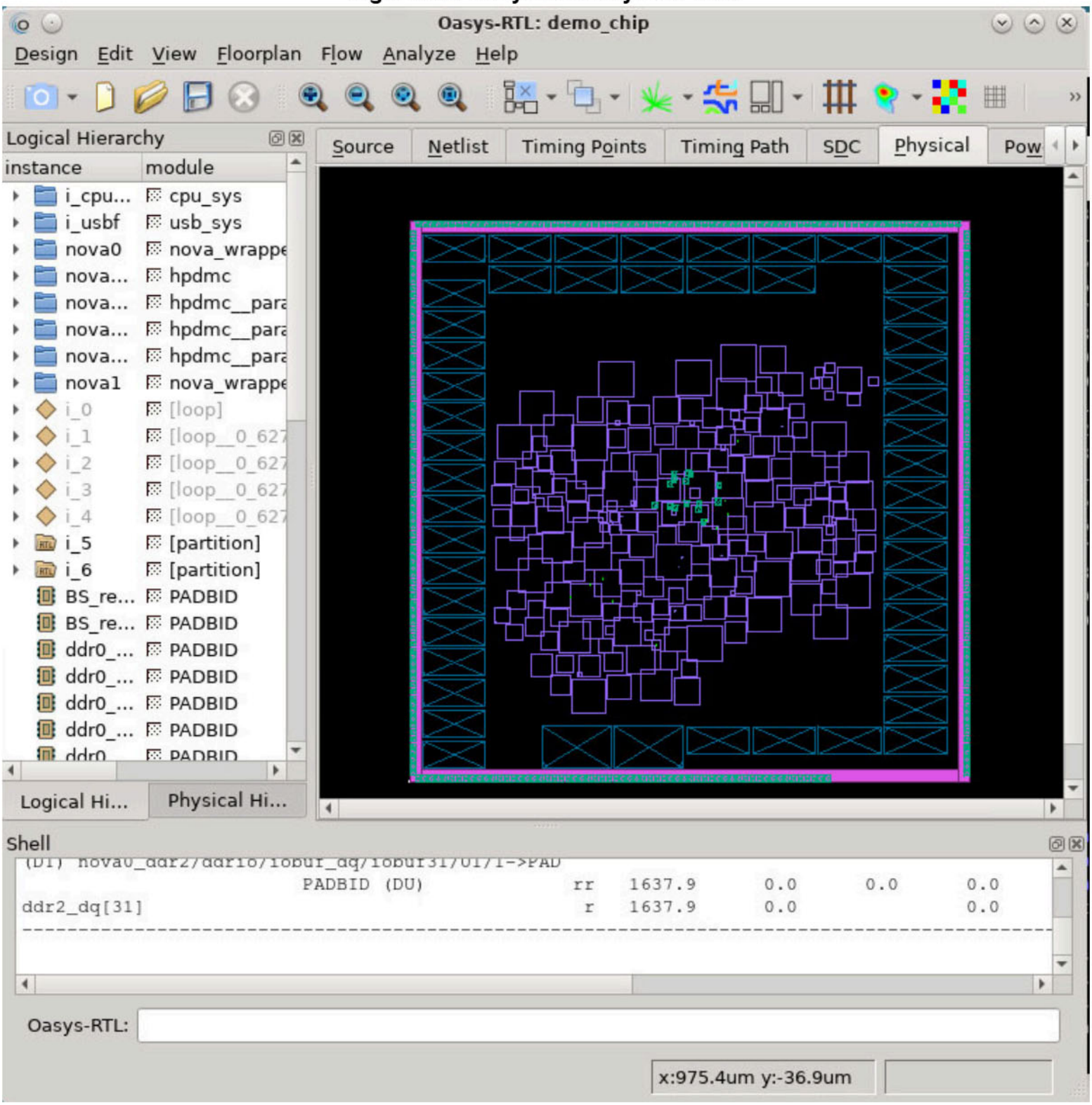
Oasys may report an error because the Source window can not find a hard-coded path to the RTL source in the ODB file. As a result, you will not be able to view or cross-probe to the RTL source but this ODB file will allow you to complete the other steps in the tutorial.

Figure 3-1. The Oasys-RTL GUI



3. Click the **Physical** tab above the source netlist view to view the initial floorplan.

Figure 3-2. Oasys-RTL Physical View



The physical view shows the placement of macros and RTL partitions. The Oasys-RTL tool starts with an empty DEF view with only the design pins placed. The tool performs floorplanning by placing the pins, macros, and RTL partitions in an optimal manner.

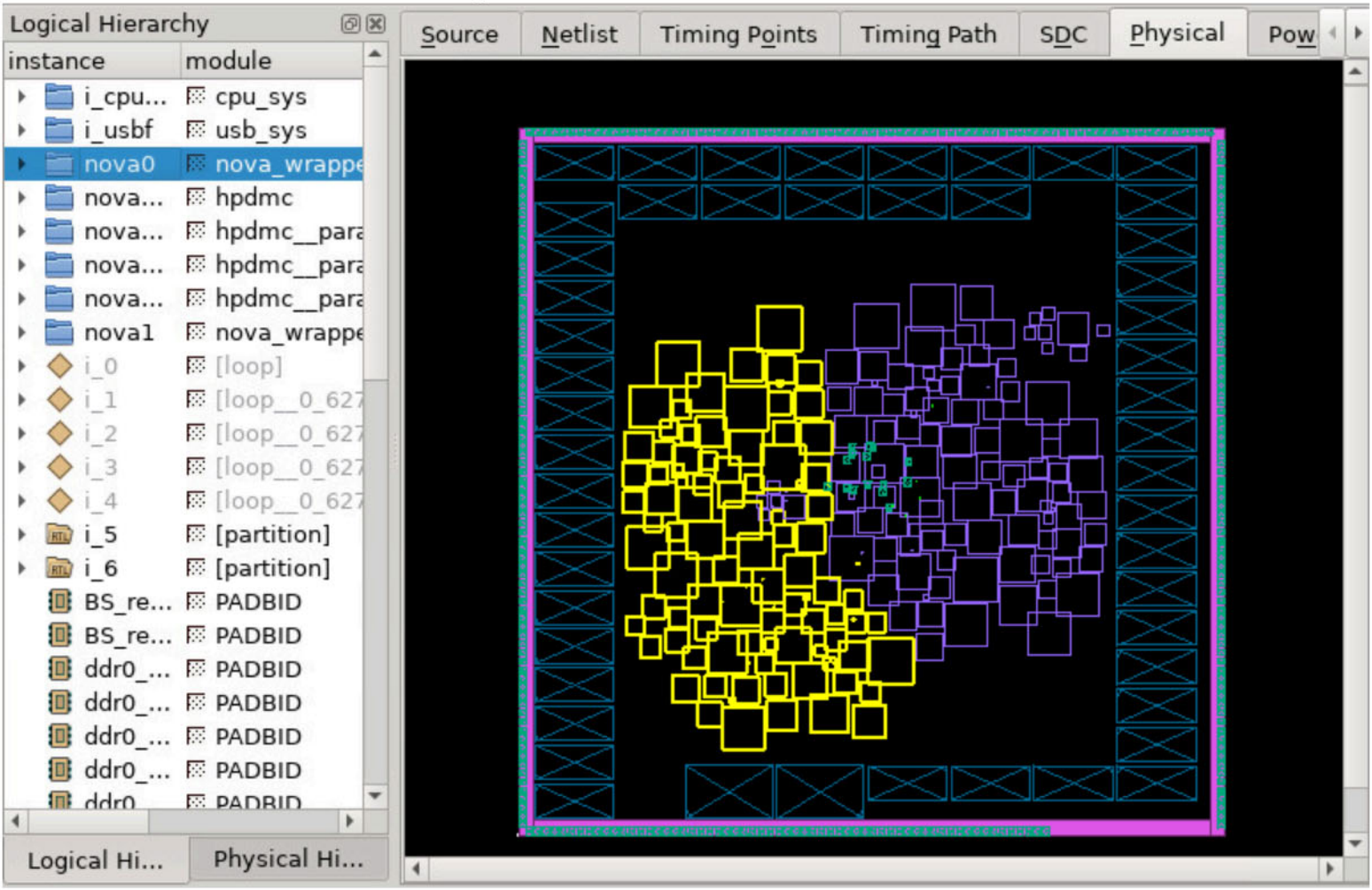
RTL partitions are displayed in purple. Macros are displayed in blue.

Note

The layout depicted in the screenshots may be different from what you see in the tool.

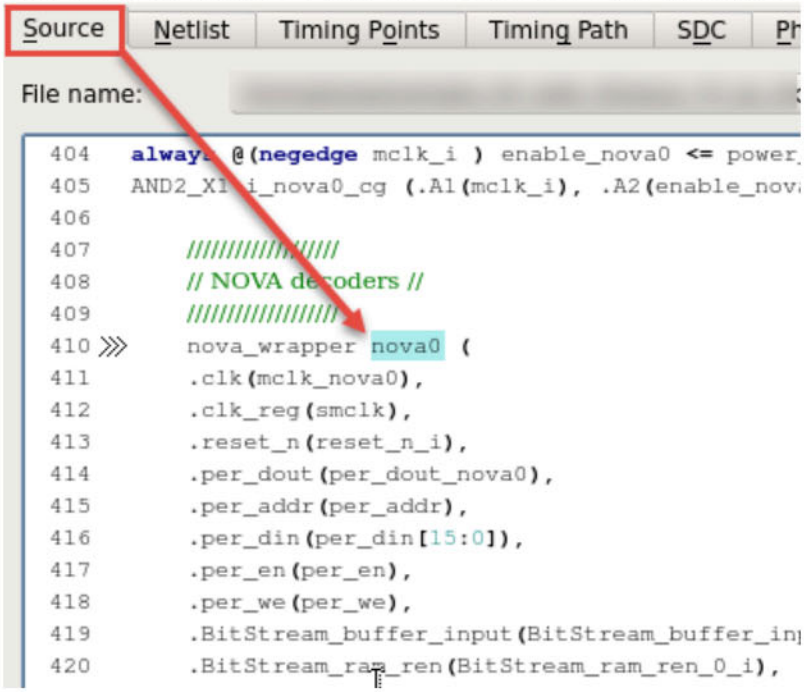
4. Select the "nova0" instance in the Logical Hierarchy browser on the left of the GUI. All child partitions that belong to the nova0 instance are highlighted:

Figure 3-3. Selected nova0 Instance



5. Click the **Source** tab. If you downloaded the RTL source from the Open Cores site, the Verilog module instantiation is also highlighted:

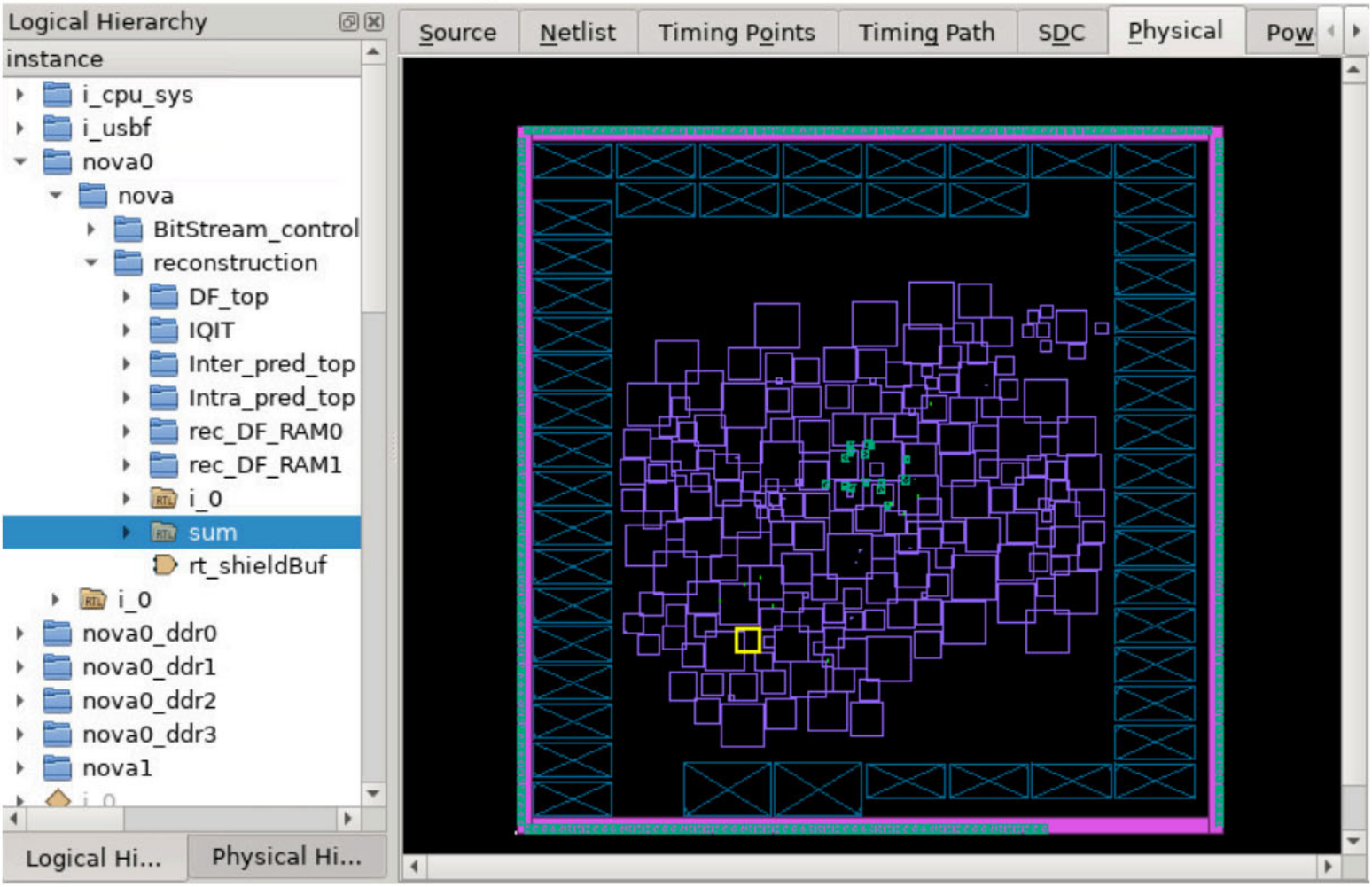
Figure 3-4. Source View of Selected nova0 Instance



The Oasys-RTL GUI provides cross-highlighting between many different design scopes, which enables you to quickly debug issues. By default, the Source tab is included in the list of design views. You can also move it to a new pane within the GUI by right-clicking on the Source tab and selecting the **Dock Tab** item. You can place a docked pane back in the main pane by right-clicking on the tab and selecting **Tabify Tab**.

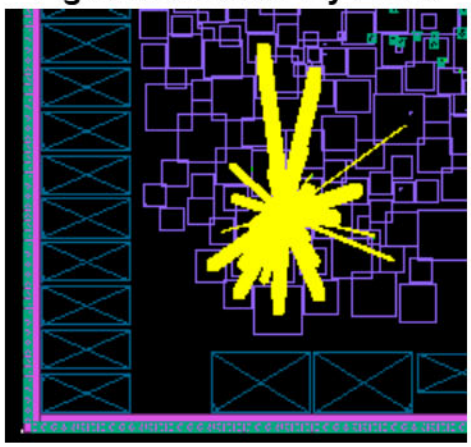
6. In the Logical Hierarchy browser, select the **nova0 > nova > reconstruction > sum** RTL partition and click the **Physical** tab.

Figure 3-5. Selected sum Instance



7. To show or hide fly lines, right-click in the physical view and choose **Fly Lines > Show Fly Lines** or **Fly Lines > Hide Fly Lines**. The fly lines show all objects that share connectivity with the highlighted instance.

Figure 3-6. Show Fly Lines



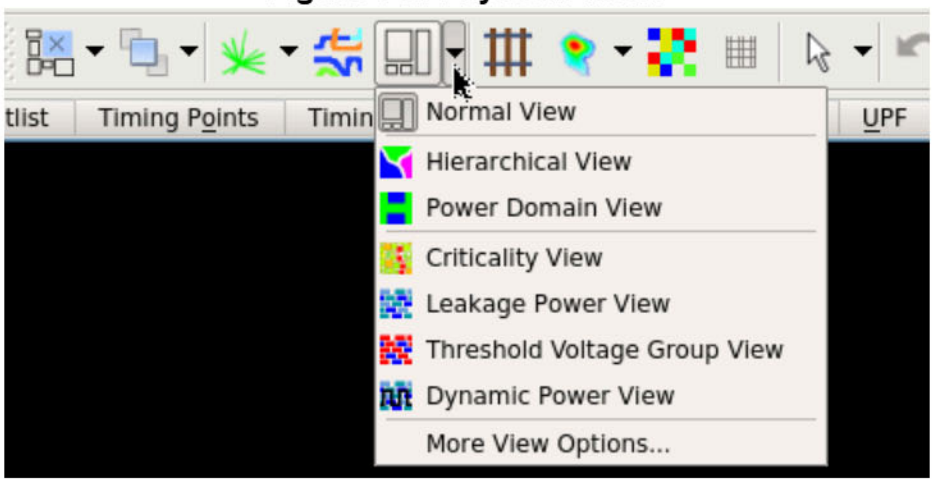
You can also see flylines by selecting an instance and clicking on the **Show Fly Lines** icon in the GUI. This icon also provides a drop-down menu that allows you to control which destination objects (eg. ports, partitions) are included when generating the fly lines.

Tip

You can also create flylines by clicking the middle mouse button on any instance.

8. Hide fly lines by right-clicking anywhere in the physical view and choosing **Fly Lines > Hide Fly Lines**.
9. Left-click anywhere outside of the core area in the physical view to unselect all highlighted instances. This returns the hierarchy selection to the top module.
10. Click the dropdown arrow next to the **Normal View** icon ( ):

Figure 3-7. Physical Views

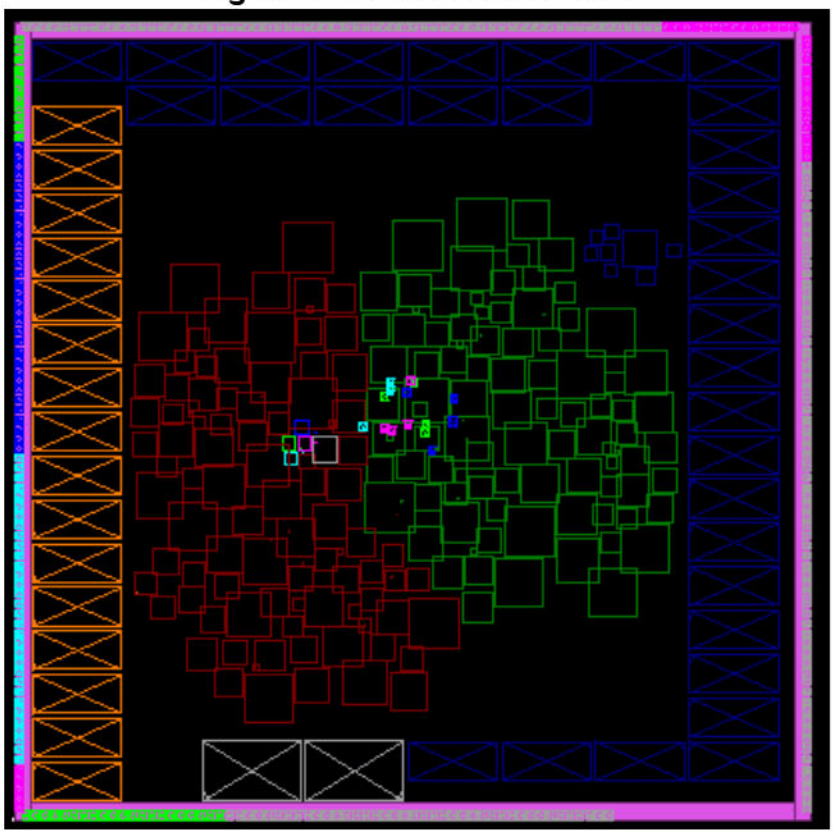


The expanded View menu allows you to control the properties that are currently displayed in the **Physical** tab.

The **Normal View** selection returns the physical view to the default setting.

11. Enable the **Hierarchical view** ( ) from the View list.

Figure 3-8. Hierarchical View



12. While the hierarchy view is active, select an instance from the Logical Hierarchy browser and note that each instance hierarchy has a unique color.

    Viewing the placement of the RTL instances in hierarchical view is useful to help understand how the design data flow is dependent on the physical placement.

13. (optional) If you plan to continue with the next procedure, do not exit from the design session.

Tip

You can close the Oasys-RTL GUI by typing stop_gui at the command prompt or using the Ctrl-. (Ctrl-period) keyboard shortcut.

Related Topics

Analyzing Timing

Analyzing Congestion

Analyzing Power

Analyzing Design for Test (DFT)

Analyzing Timing

View timing issues in the example design.

Procedure

1. If not already running, invoke the Oasys-RTL GUI and source the *scripts/open_database.tcl* script to load the demo chip design ODB databases.
2. To generate a report of the longest timing paths, type **report_timing** at the command prompt.

   The report indicates various details of a path by suffixing symbols to the end of the library cell on the path. The following symbols are used to indicate the timing attributes:

   - * — An output has been virtually buffered for timing.
   - # — An output, which is typically a register, has been upsized for loading.
   - DU — A library cell has the "dont_use" property set to true.
   - DT — A cell has the "dont_touch" property set to true.
   - retime borrow — A state cell has a retime constraint and timing was borrowed.

   The following example shows an instance with a virtual buffer delay (denoted with an *) within a timing path:

   ```
   b0/sreg[2]/CK->Q SDFFR_X1_LVT#* rr 75.9 75.9 75.9 0.0 0.0 2.7 33.1 4
   483, 815 /PD_TOP (0.85)
   ```

3. If the design currently yields no negative slack paths, you can reduce the clock period slightly to produce some timing violations to analyze in the following subsection of the tutorial.

   To reduce the clock period

   a. Open the SDC constraint file at *constraints/demo_chip_func.sdc*

   b. Change the **clock_period** variable from 100 to 99.5.

   c. Save the changed SDC constraint file.

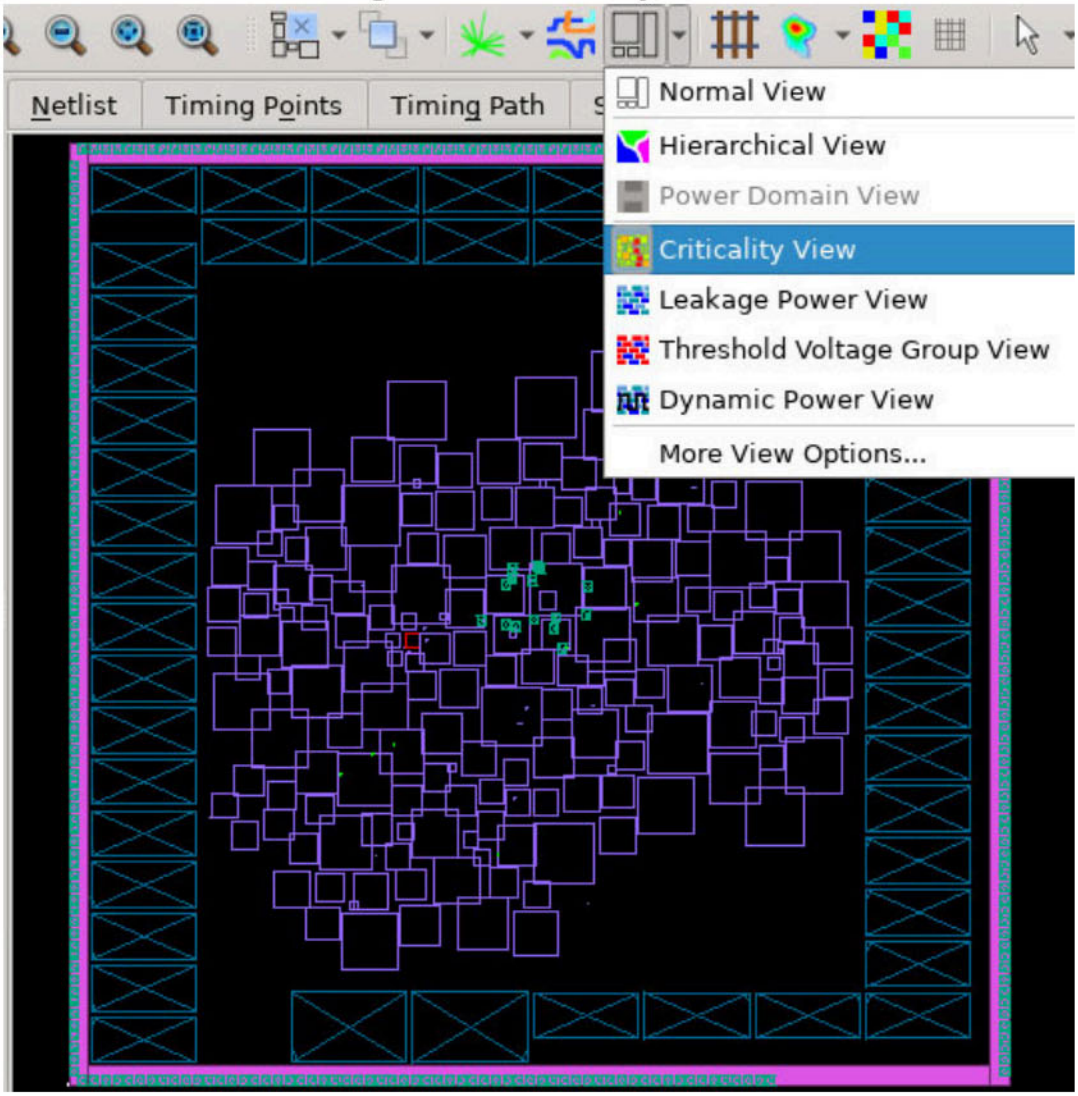
   d. Source the SDC file in Oasys to update the clock constraints in the design:

      source constraints/demo_chip_func.sdc

   You can verify the clock periods used for timing analysis using the *report_clocks* command.

4. Click the **Physical** tab and choose **Criticality View** () from the Vie**w dr**opdown list to visualize timing violations across the design.

Figure 3-9. Criticality View



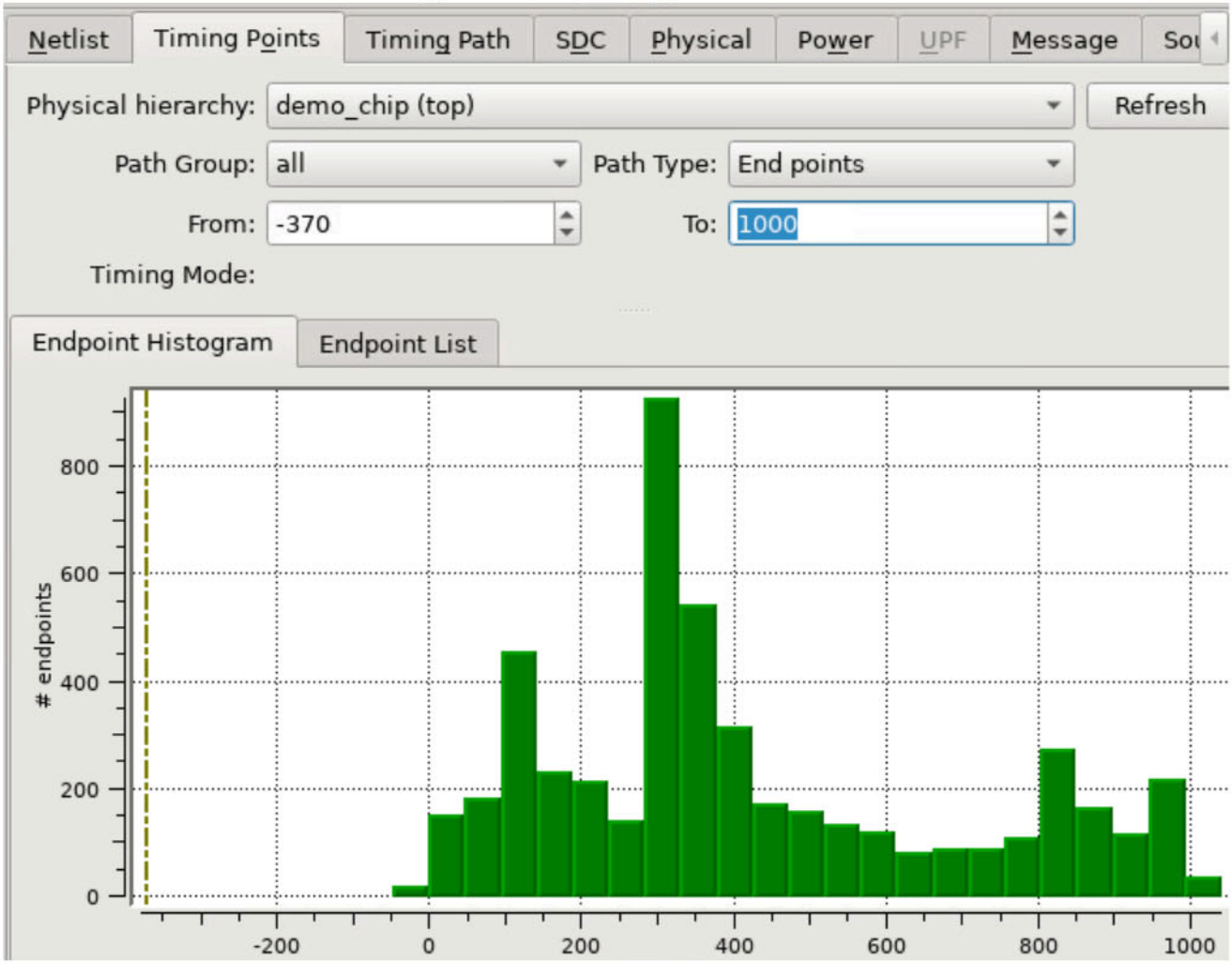
This view shows the instances with timing violations within the placed hierarchy. It can provide a high-level perspective of the violations before you analyze the timing reports or make design modifications.

Tip

If only part of the design was highlighted, you are only viewing the criticality for selected instances. Left-click outside of the core area in the Physical View to return the selection to the top level, click the **Normal View** button, and then choose the **Criticality View** again to highlight the entire design.

5. Click the **Timing Points** tab. A histogram of the timing slack is displayed. To provide a better representation of the number of endpoints that were close to violating timing (eg. negative slack), the maximum value in the histogram (To:) was changed from the default value to 1000ps.

Figure 3-10. Timing Points Tab

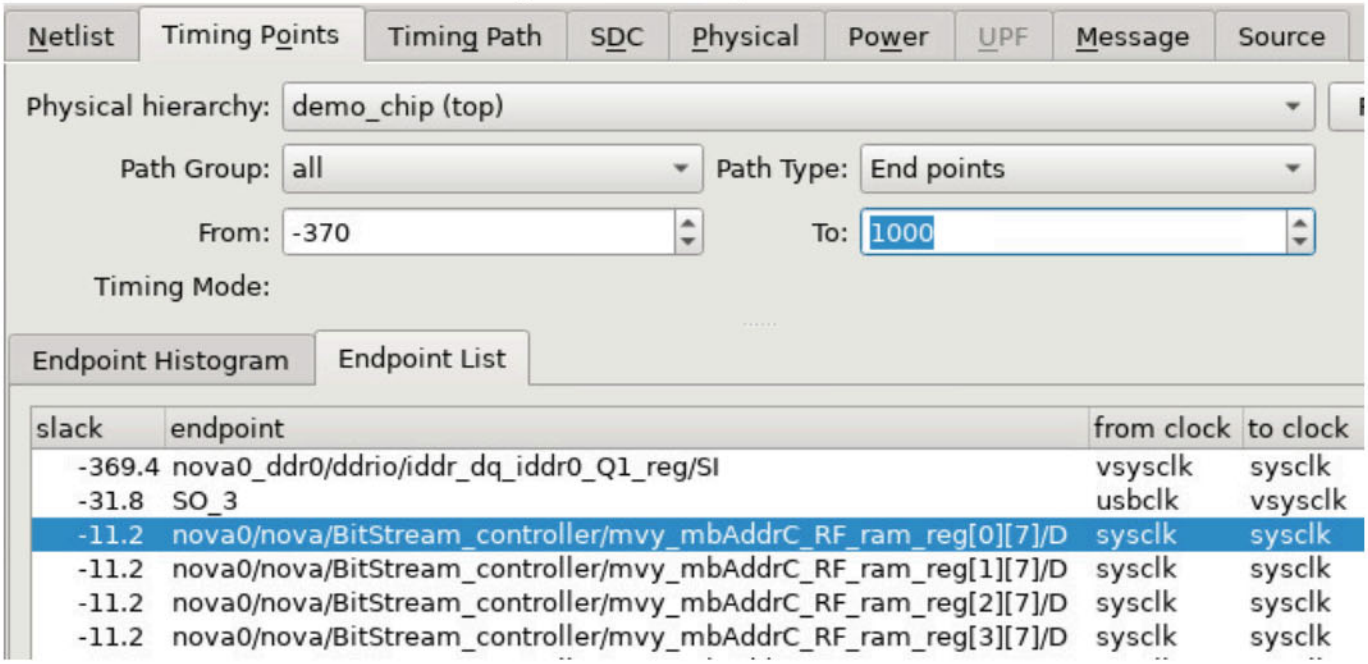


6. Click the **Endpoints List** tab (within the **Timing Points** tab) to view a table of slack and endpoints.
7. Select the first value at the top of the table.

Note

The paths and slack values in the following figures may not match your results.

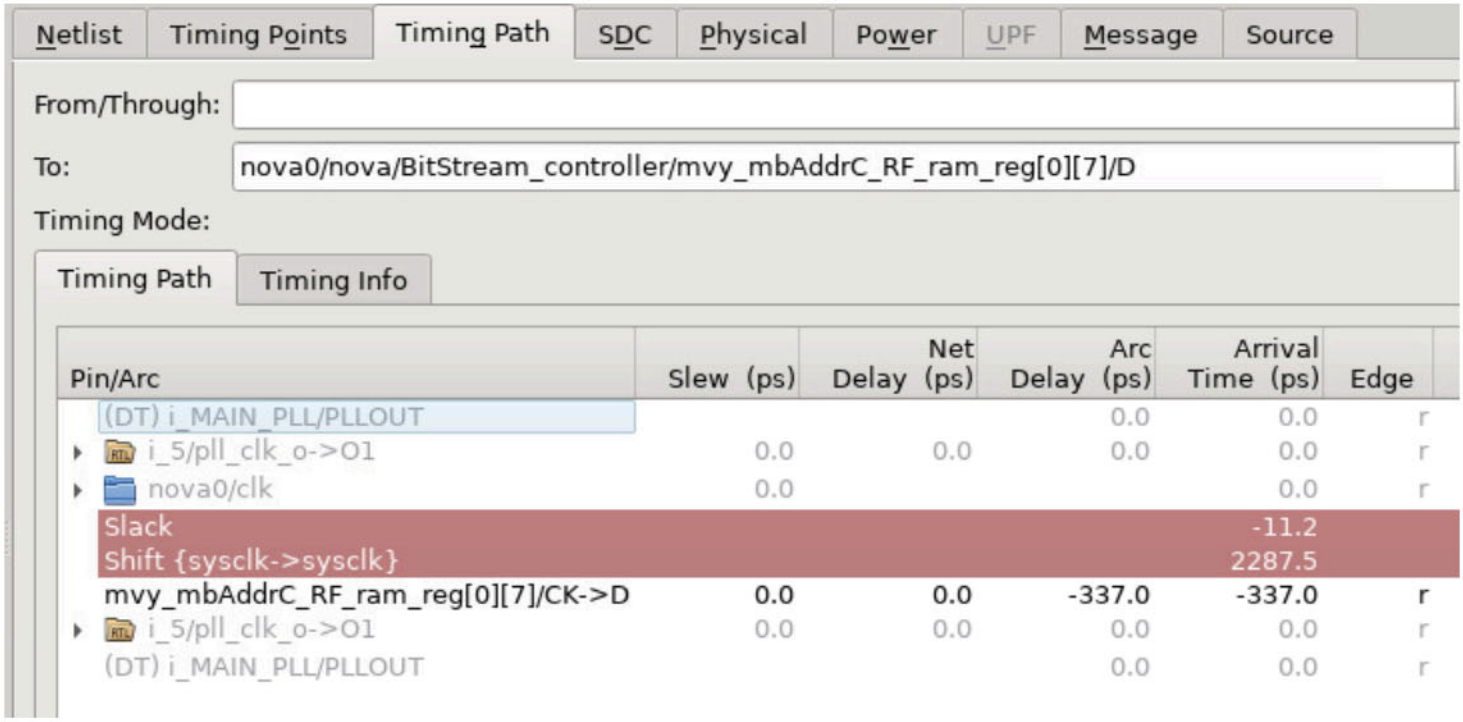
Figure 3-11. Endpoint List



8. Click the **Timing Path** tab.

   Another **Timing Path** tab appears below. This tab displays the timing start and end points for the path you selected in the previous step.

Figure 3-12. Timing Path



9. Click the **Timing Info** tab. This view displays the originating and receiving elements of the timing path in addition to timing metrics such as the clock period, uncertainty, latency, and setup time.
10. Click the **Timing Path** tab and click the **Open to Rtl** button at the bottom of the window. This feature expands the selected timing path to the RTL level of hierarchy without showing low-level mapped technology cells in the timing path.

    Some common timing symbols that appear in the Timing Path window are described in Table 3-2.

Table 3-2. GUI Timing Symbols

Symbol	Identifier	Description
	RTL partition	RTL partition in the design. These symbols indicate highly interconnected blocks that are placed in close proximity.
	<register_name>	Starting point of the timing path.
	Mux symbol with notch at inputs	Datapath in the timing report.
	Mux symbol	Mux implementation in the timing path.
	AND symbol	Combinational logic in the timing path.
	always_<name>	RTL “always” block.

11. Explore the cross-highlighting features of RTL components in the Timing Path tab by clicking on elements in the Timing Path and viewing the element in the Physical window. If you downloaded the RTL source for the lower level blocks, you can also view the RTL code for the element in the Timing Path in the Source window.
12. (optional) If you plan to continue with the next procedure, do not exit from the design session.

Related Topics

Using the Oasys-RTL GUI

Analyzing Congestion

Analyzing Power

Analyzing Design for Test (DFT)

Analyzing Congestion

View congestion issues in the example design.

Procedure

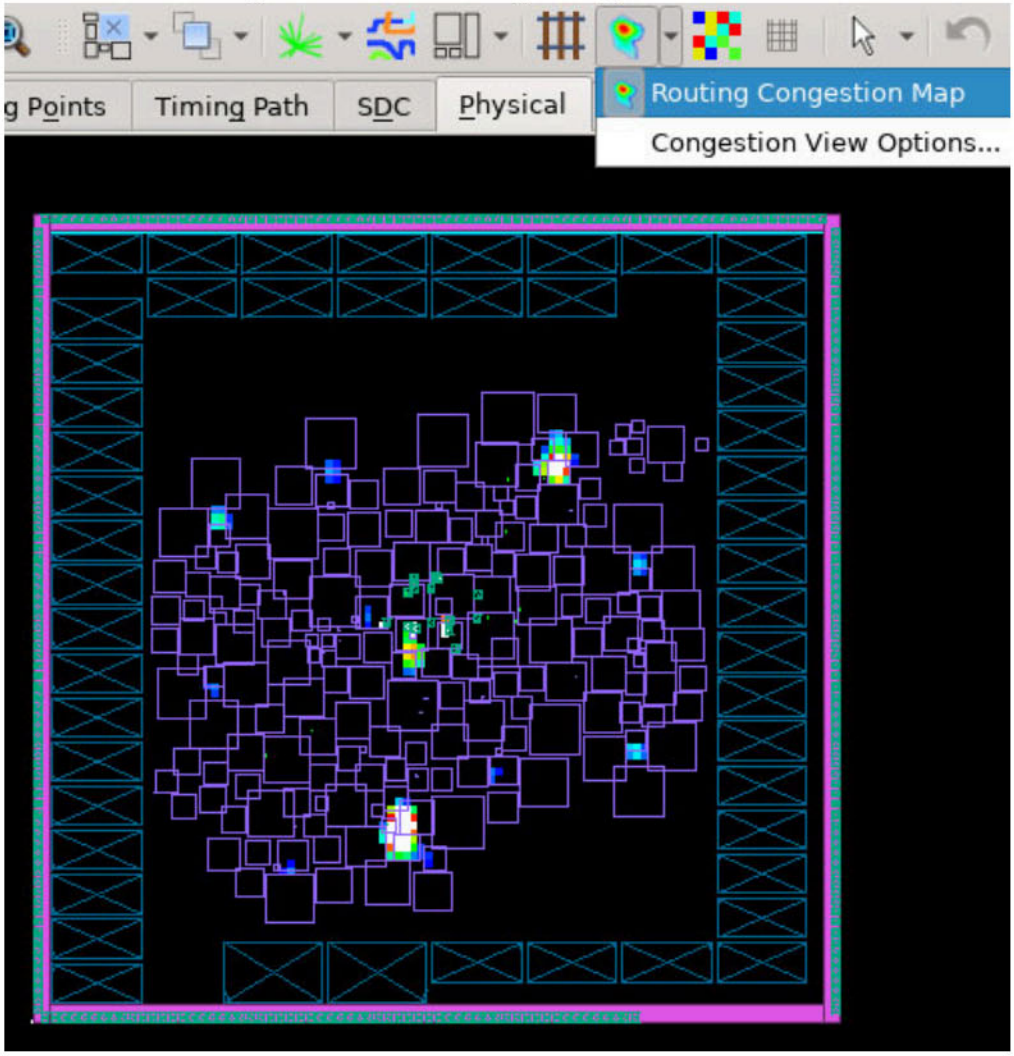
1. If not already running, invoke the Oasys-RTL GUI and source the *scripts/open_database.tcl* script to load the synthesized design.
2. Select the **Physical** tab and press Ctrl + 0 to fit the core area to the current view.
3. Left-click outside of the core area to deselect any selected instances.
4. Right-click outside of the core area and select **View > Normal View** to enable Normal view.
5. Select the **Routing Congestion Map** icon ( ) in the main menu.

   This plots a congestion map over the physical design (Figure 3-13).

Note

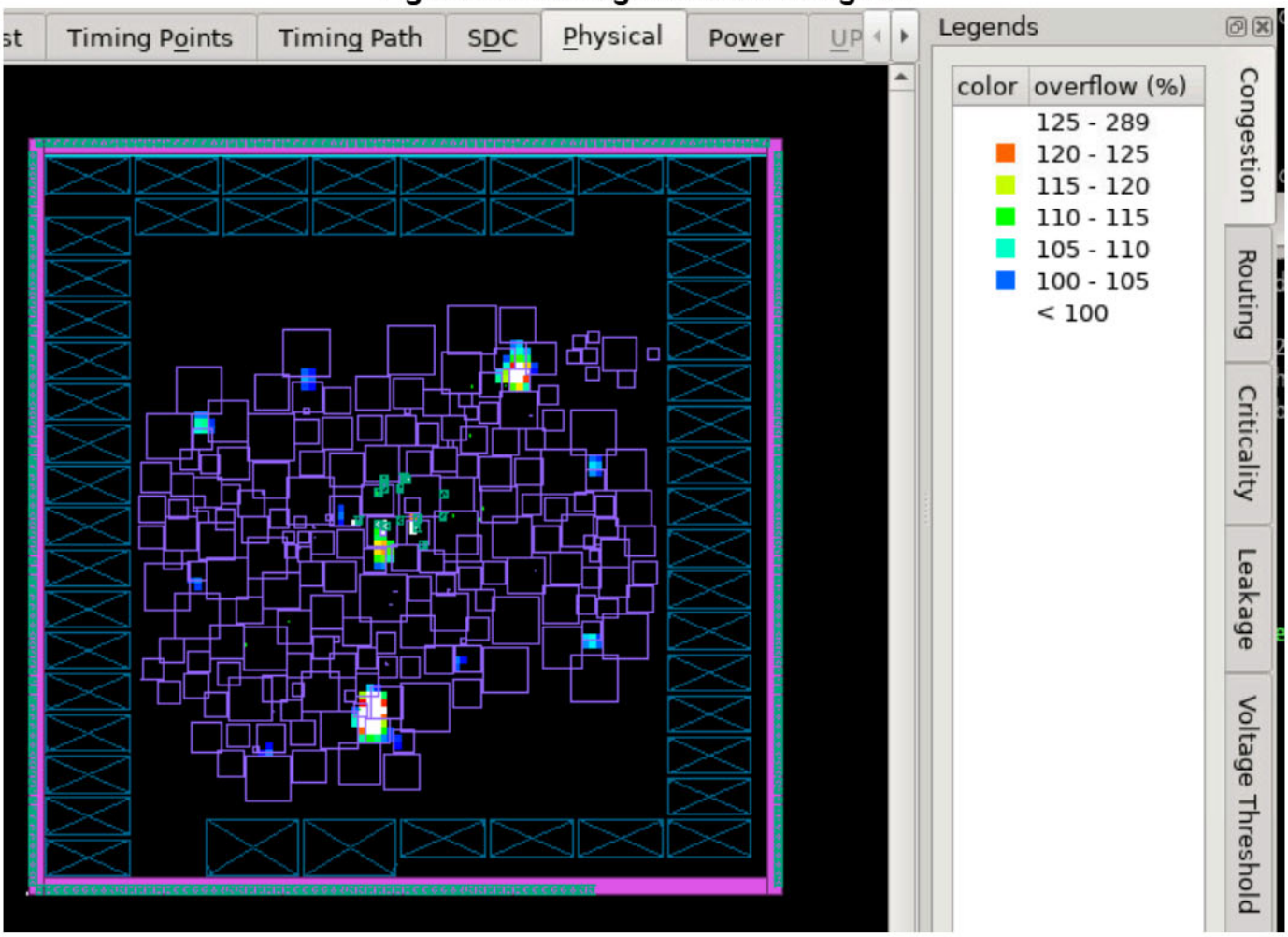
By default, the congestion map displays in "grid view". To disable this (as shown in figure), select the dropdown list arrow beside the **Routing Congestion Map** icon and select Congestion View Options. Then in the Congestion Set dialog box, uncheck Grid View and click **Apply**.

Figure 3-13. Routing Congestion View



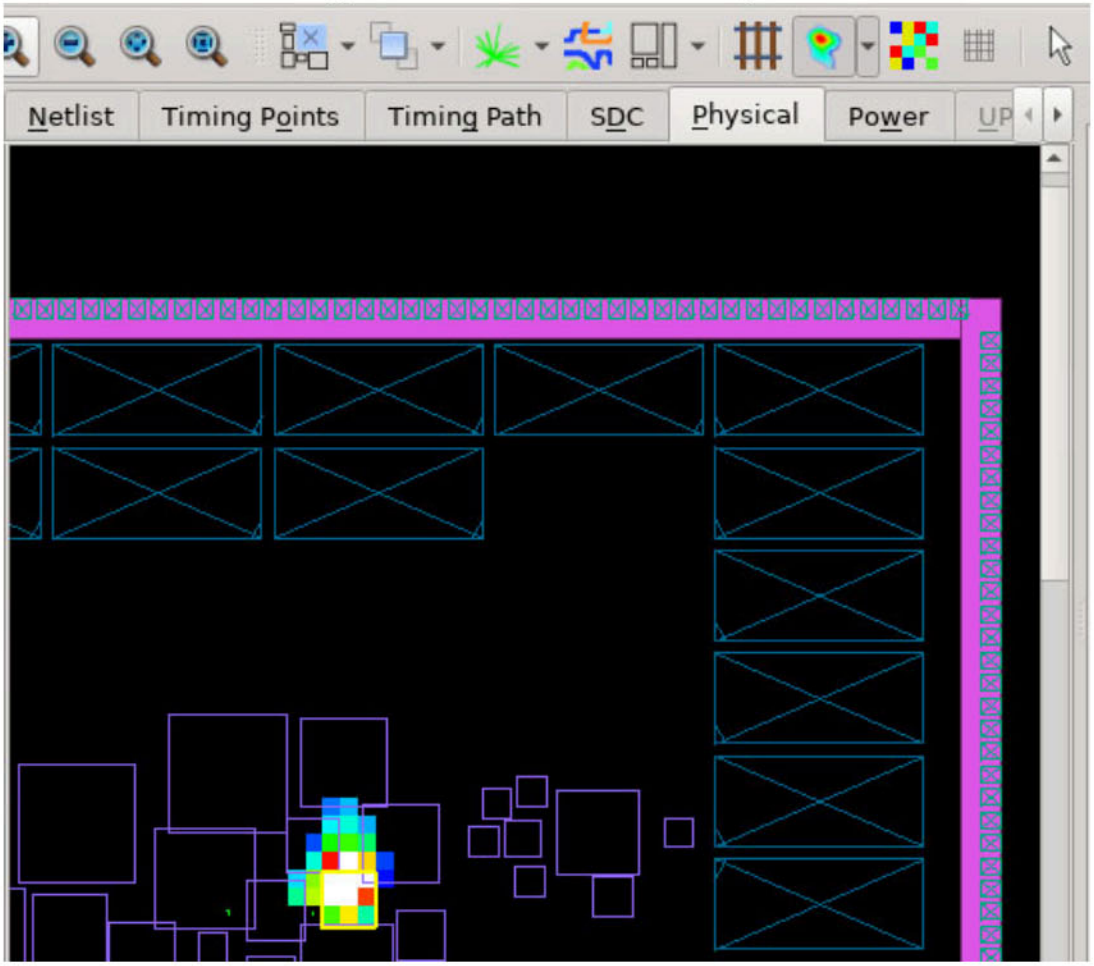
6. Select **View > Legends** from the main menu. The Congestion tab that opens on the right side of the main window defines how the colors map to the congestion overflow percentage.

Figure 3-14. Congestion View Legend



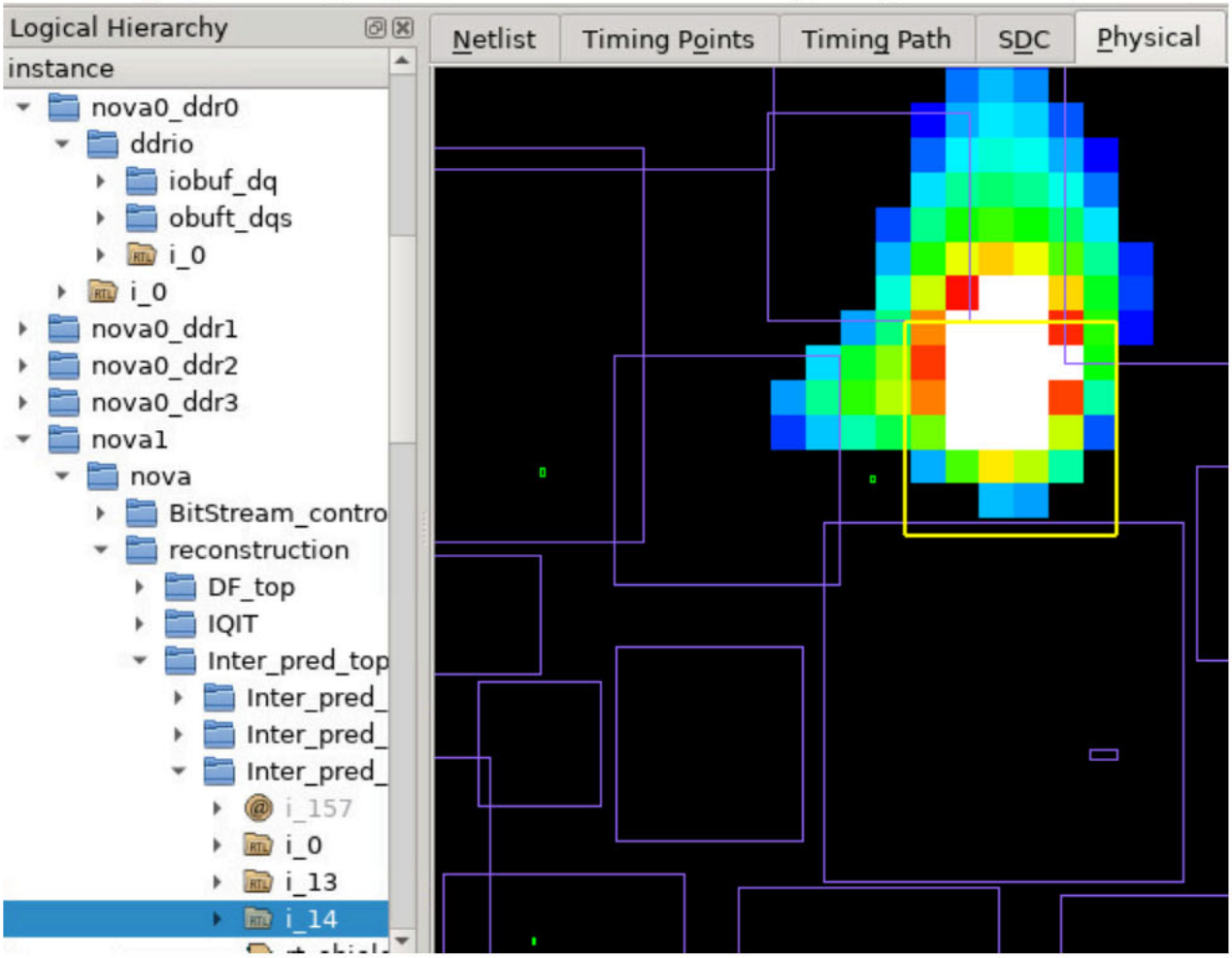
7. Click the **Select** icon.
8. Left-click and drag a box to zoom to a congestion hotspot in the physical view.

Figure 3-15. Zoom in on Hotspot



9. Left-click the instance that represents the source of the congestion hotspot. Observe the corresponding highlight in the Logical Hierarchy browser.

Figure 3-16. Highlighted Instance After Clicking Congestion Hotspot



10. (optional) If you downloaded the OpenCore source files, click the **Source** tab.

    The RTL code that generates this congested area is highlighted in the Source view. This is used to understand the source design components that cause congestion issues in the Physical view.

11. (optional) If you plan to continue with the next procedure, do not exit from the design session.

Related Topics

Using the Oasys-RTL GUI

Analyzing Timing

Analyzing Power

Analyzing Design for Test (DFT)

Analyzing Power

View the physical power properties of the design.

Procedure

1. If not already running, invoke the Oasys-RTL GUI and source the *scripts/open_database.tcl* script to load the synthesized design.
2. Select the **Physical** tab and use Ctrl+0 to fit the core area to the current window.
3. Click the **Routing Congestion Map** icon to unselect congestion view (if it is still showing from the last task).
4. Left-click outside of the core area to deselect any selected instances.
5. Select **View > Legends** from the main menu. The Domain tab that opens on the right side of the main window defines how the colors map to the specified power domains in the design.
6. Enter the following command at the Oasys-RTL command prompt:

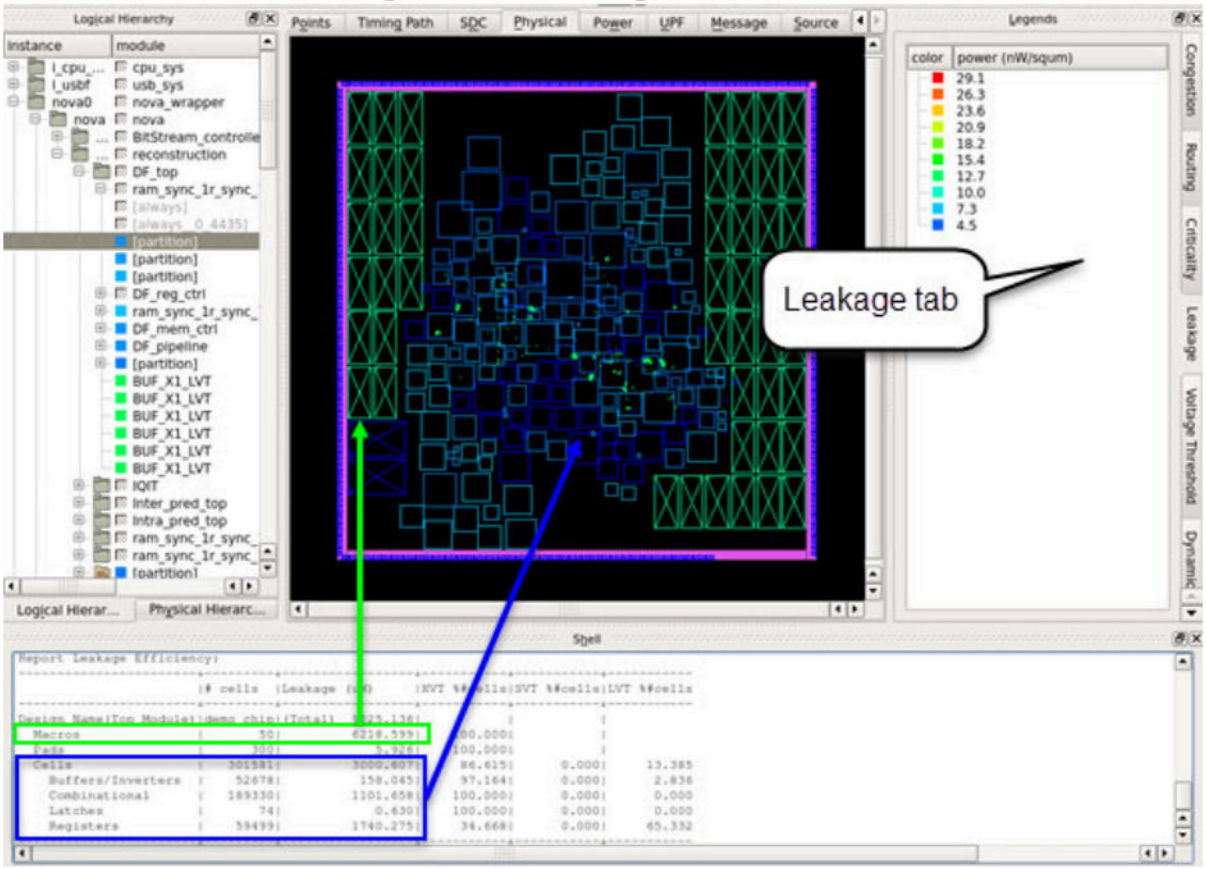
   report_leakage

   Note that the Macros row in the leakage report consumes significantly more static power than the leaf cells (Cells row):

```
Report Leakage Efficiency:
----------------------+---------+-----------------+-------------------
                      |# cells  |Leakage (uW)     |default_vth %#cells
----------------------+---------+-----------------+-------------------
Design Name(Top Module)|demo_chip|(Total)  9274.380|
  Macros               |       50|         6218.599|            100.000
  Pads                 |      300|            5.926|            100.000
  Cells                |   301317|         3049.856|            100.000
```

7. Select the **Leakage Power View** () from the View dropdown list.

Figure 3-17. Leakage Power View



The leakage map plotted on the design visualizes the report, where the macro blocks dissipate higher amounts of static power than the placed cell area.

The **Leakage** tab of the Legends window maps the colors to the leakage power per square micron.

8. Enter the following command at the Oasys-RTL command prompt:

   report_power

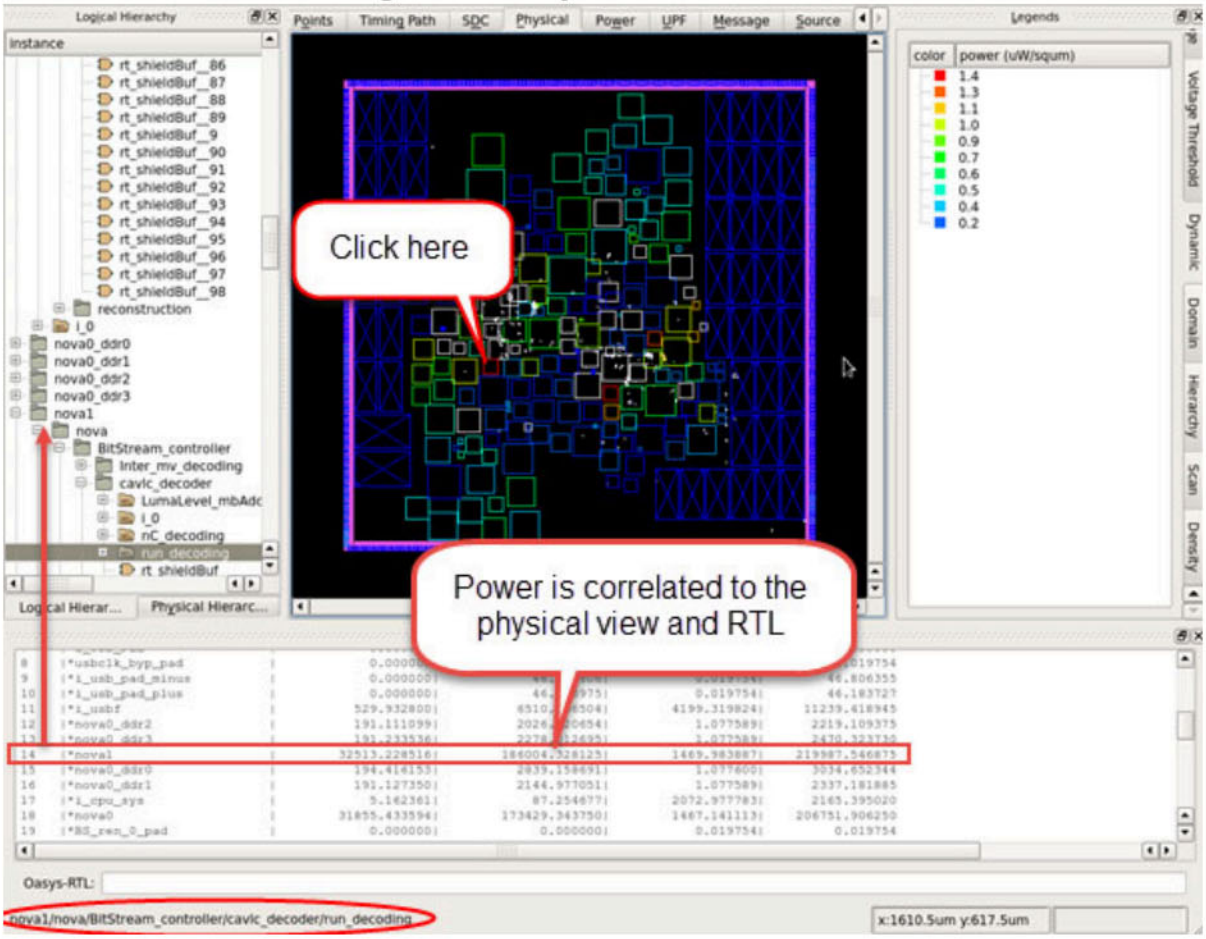
   This itemizes the dynamic and leakage power for all design blocks. This command takes some time to execute.

9. Select the **Dynamic Power View** ( ) from the View dropdown list.

   The **Dynamic** tab of the Legends window maps the colors to the dynamic power per square micron.

10. Click one of the red instances in the core area in the Physical view.

Figure 3-18. Dynamic Power View



Note that this instance belongs to the nova1 block, which has the highest reported switching power in the report.

11. Click the **Source** tab and note that instance is also highlighted in the RTL code.
12. (optional) If you plan to continue with the next procedure, do not exit from the design session.

Related Topics

Using the Oasys-RTL GUI

Analyzing Congestion

Analyzing Timing

Analyzing Design for Test (DFT)

Analyzing Design for Test (DFT)

View scan chains and debug DFT issues in the example design.

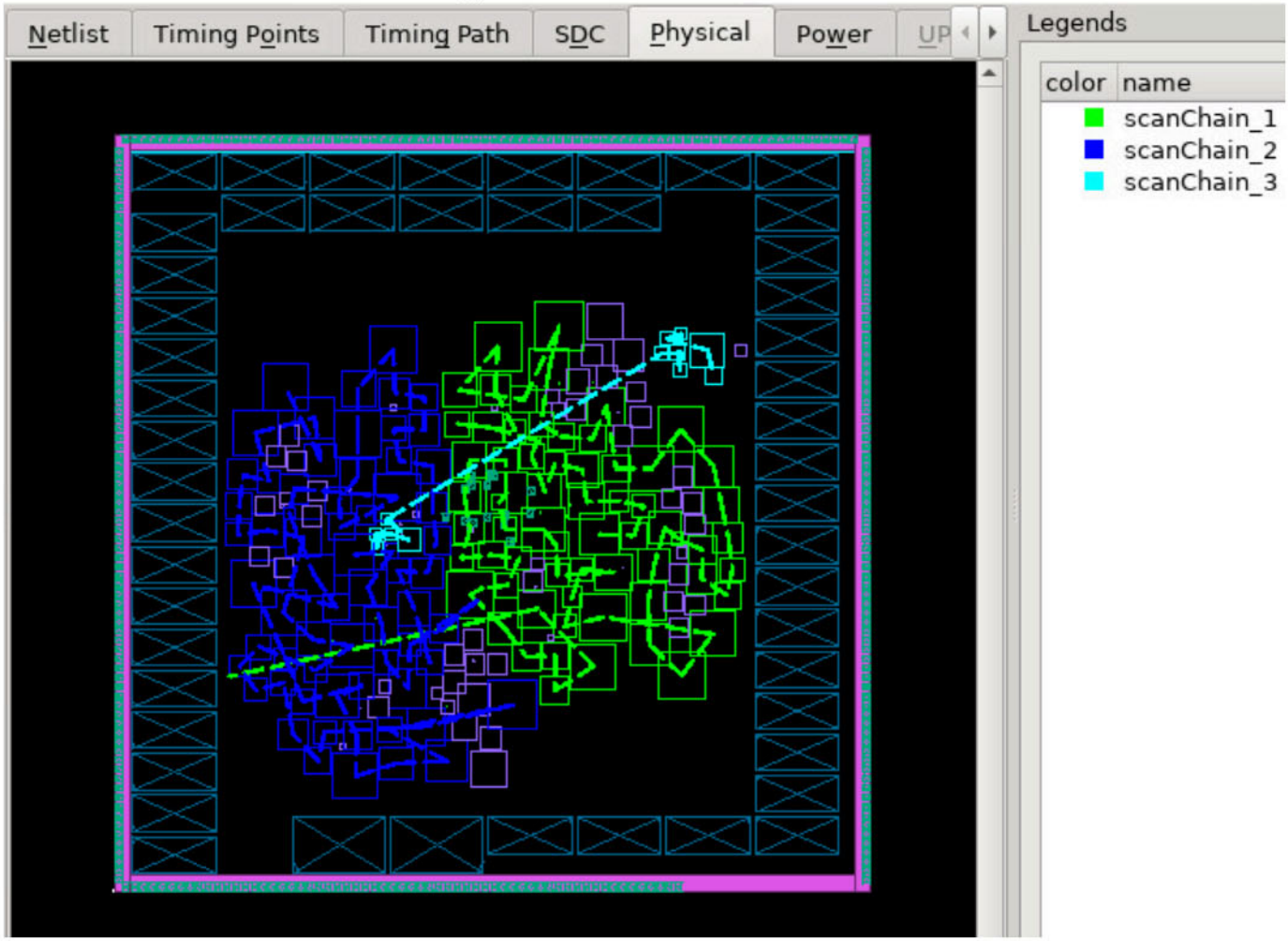
Procedure

1. If you are not continuing from the previous session, invoke the Oasys-RTL GUI and source the *scripts/open_database.tcl* script to load the synthesized design.
2. If you did not insert Scan Chains into the design yet, you run either of the following scripts. For this design, scan insertion should take approxiately10 minutes to complete.

   (Using Oasys DFT) : *source scripts/oasys_dft.tcl*

   (Using Tessent) : *source scripts/oasys_tessent_dft.tcl*
3. Select the **Physical** tab, and select **Normal View** from the View dropdown list.
4. Click the **Scan Chains** icon ( ) in the main menu ().Figure 3-19

Figure 3-19. Scan Chain View



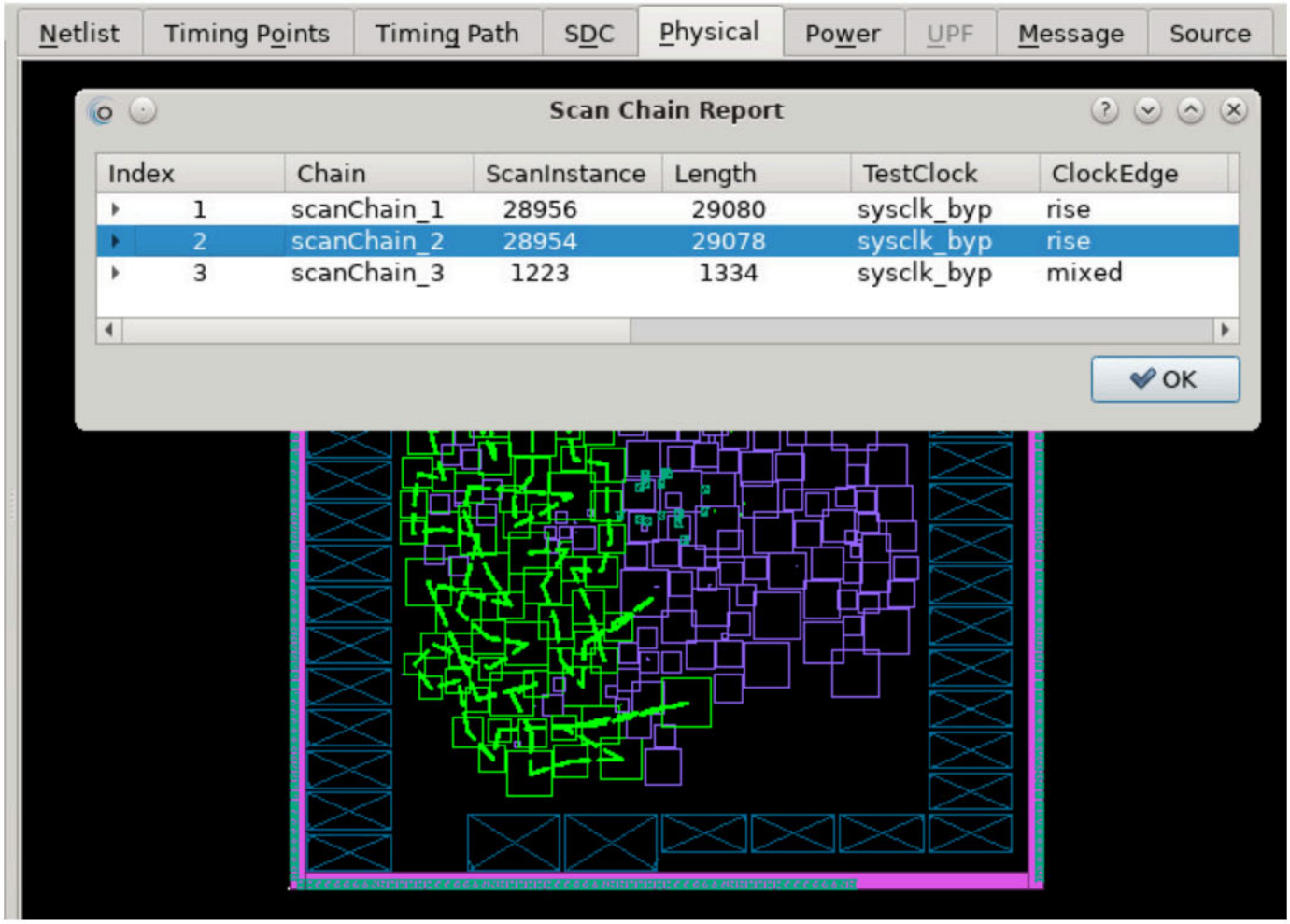
Each scan chain path is drawn with a unique color.

5. Select **Analyze > Test > Report Scan Chains** from the main menu.

This opens the Scan Chain Report dialog box.

6. Click scanChain_2. This highlights only the selected scan chain (Figure 3-20).

Figure 3-20. scanChain2 Highlighted



7. Close the Scan Chain Report window.
8. Run the following set of commands at the command line prompt.

Note

These command are necessary because the ODB previously loaded by the *scripts/open_database.tcl* script already had DFT violations fixed.

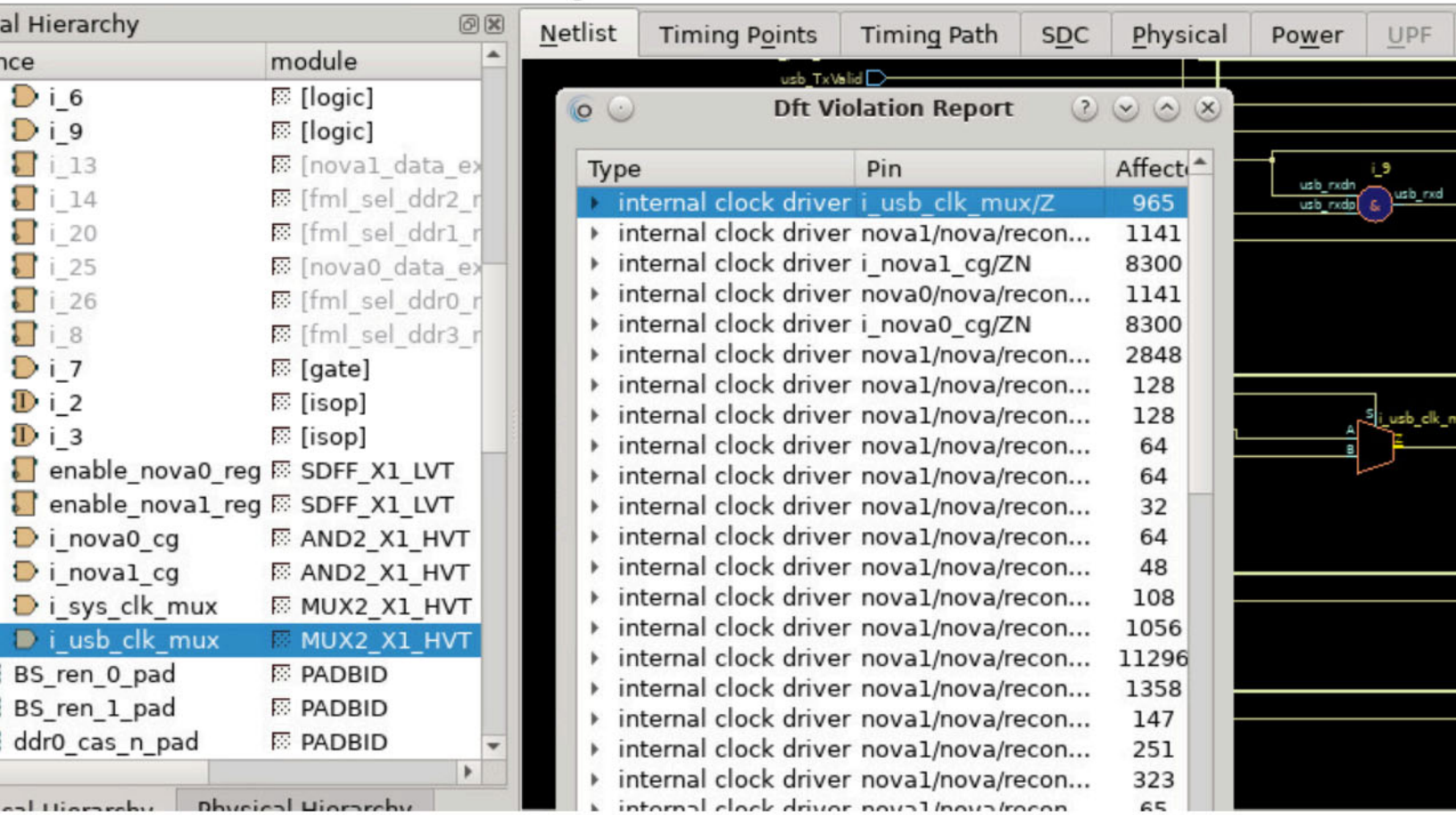
```
# <dft> is oasysdft or tessent
delete_design
read_db output/odb/demo_chip.<dft>pre_fix.odb
check_dft
```

9. Click the **Netlist** tab.
10. Choose **Analyze > Test > Report Dft Violations** from the main menu.

This opens the Dft Violation Report dialog box (Figure 3-21).

11. Select the first violation in the report. The machine-generated instance names may be different in your design.

Figure 3-21. DFT Violations



The **Netlist** tab displays the DFT violation in the context of the system schematic. You may need to zoom in to see the problem. In this case, the mux blocks the clock signal from propagating to the USB PHY.

To address this issue, you could fix the RTL code or use the autofix feature within the tool. The Oasys-RTL tool has the capability to fix the clock and DFT DRC violations with the **fix_dft_violations** command. This command fixes violations by inserting test logic so that, in scan mode, the corresponding signals propagate from primary inputs and bypass the internally generated signal that is uncontrollable.

12. To use the autofix feature to address the DFT violations in the design, issue this command:

```
fix_dft_violations -type all -test_clock sysclk_byp -test_control scan_mode
```

This command inserts logic into your design to address the DFT DRC violations.

13. Close the Oasys-RTL tool and continue to Chapter 4, "RTL Floorplanning".

Related Topics

Using the Oasys-RTL GUI

Analyzing Timing

Analyzing Power

Analyzing Congestion

Chapter 4
RTL Floorplanning

The Oasys-RTL graphical user interface can be used to modify a floorplan.

Generating an Initial Floorplan

To generate an initial floorplan, set a new aspect ratio, change the core utilization, or both.

The default core area has a square aspect ratio. The aspect ratio is defined as the height to width ratio. The default core utilization is 60%. To minimize or remove potential congestion hot spots (as seen in the Congestion Map), modify these values to reach an optimum result.

Prerequisites

- A generated design database (odb).

Procedure

1. Enter the following command to invoke the Oasys-RTL GUI:

   ```
   $OASYS_HOME/bin/oasys -gui -log output/logs/floorplan.log
   ```

2. Open the synthesized and optimized Oasys-RTL database, generated in Chapter 2, "Design Synthesis", by entering the following command at the Oasys-RTL command prompt at the bottom of the GUI window:

   read_db output/odb/demo_chip.oasys_final.odb

3. At the Oasys-RTL prompt in the GUI, enter the following command:

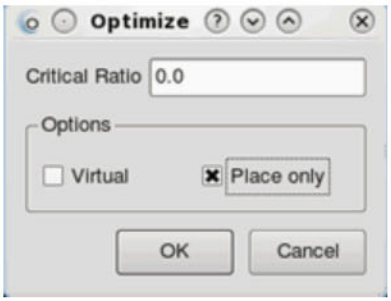
   delete_physical -all

   This completely removes the placed physical design.

4. Select the **Physical** tab. Notice that the view is empty.

5. From the menu, select **Flow > Optimize**.

This opens the Optimize dialog box.



6. Select the Place only selection box and click **OK**.

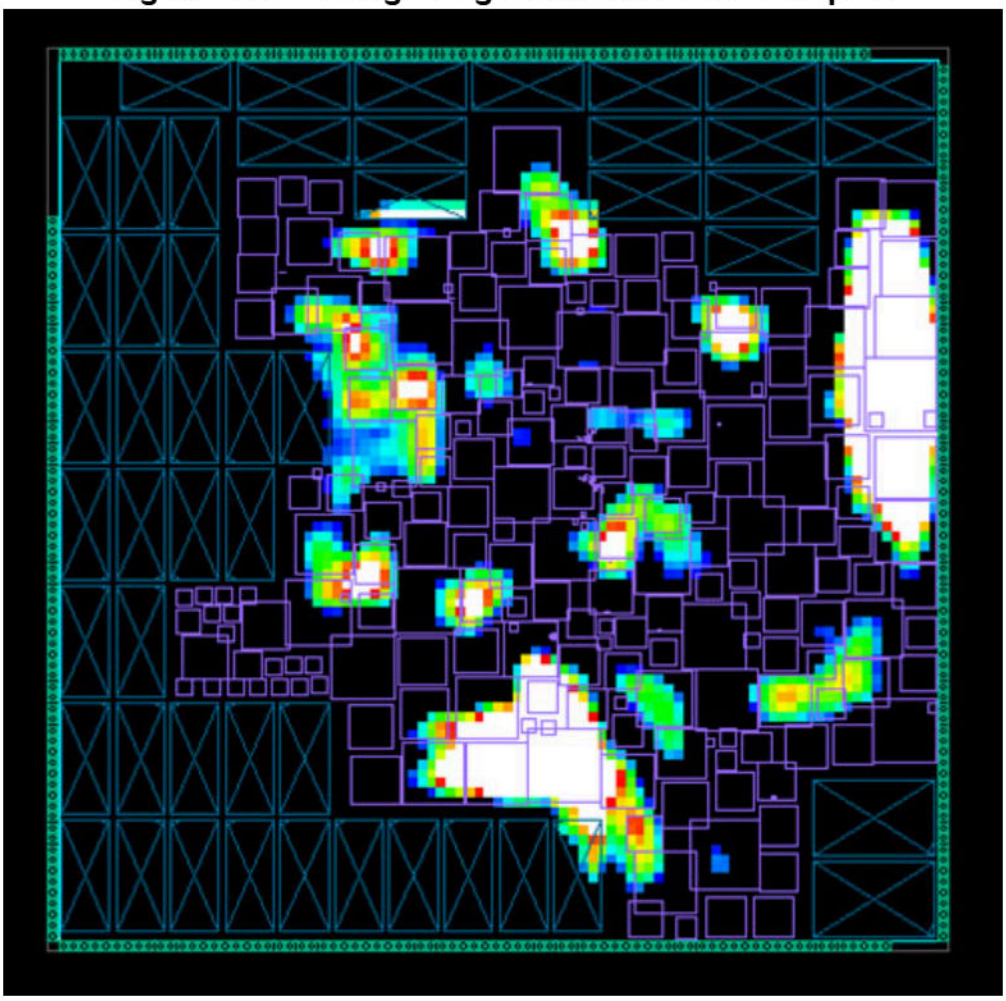
   The shell view shows the progress of the operation. At completion, the Physical view shows the placed macros and RTL blocks.

7. Click the **Routing Congestion Map** icon to see possible congestion areas in the placement (Figure 4-1).

Note

By default, the congestion map displays in "grid view". To disable this (as shown in Figure 4-1 and Figure 4-3), select the dropdown list arrow beside the **Routing Congestion Map** icon and select Congestion View Options. Then in the Congestion Set dialog box, uncheck Grid View and click **Apply**.

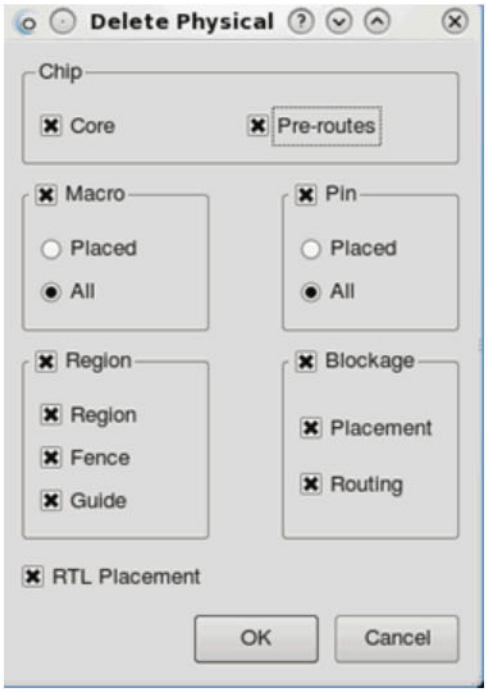
Figure 4-1. Routing Congestion View With Hotspots



Notice the hot spots throughout the design. To minimize these, modify the aspect ratio and core utilization.

8. Remove the physical data from the design. To do this, do one of the following:
   - Enter “delete_physical -all” in the Oasys-RTL prompt at the bottom of the GUI.

- From the menu, select **Floorplan > Delete Physical** and then select the Core and Pre-Routes selection boxes. All other fields are already selected.



  Click **OK**. This opens a dialog box asking for confirmation to delete all physical information. Click **Yes**.

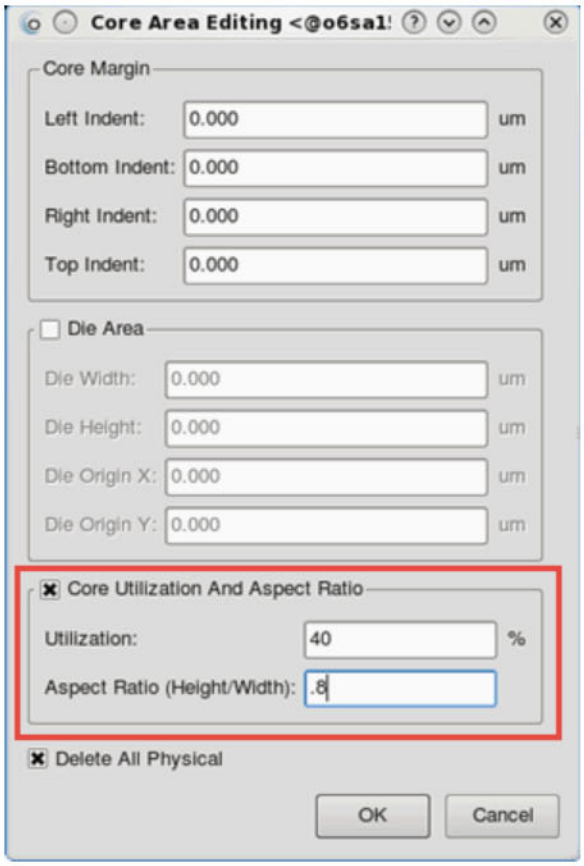
9. From the menu, select **Floorplan > Edit Core Area**.

   This opens the Core Area Editing form.

10. Select Core Utilization And Aspect Ratio and set the following values:

    - Utilization = 40
    - Aspect Ratio (Height/Width) = 0.8

Select the Delete All Physical selection box and click **OK**.

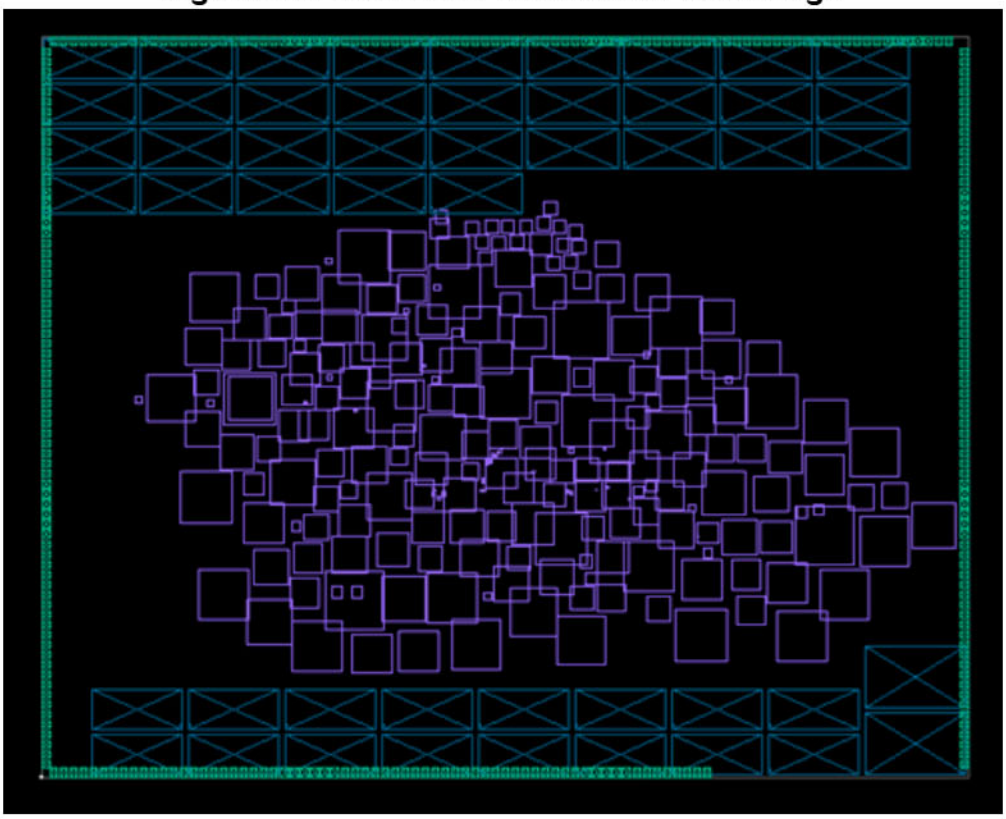


In the Physical tab, a new bounding box using the modified design aspect ratio appears. Notice that it is a rectangle.

11. To generate a new placement, select **Flow > Optimize** from the menu. In the **Optimize** dialog box, select the Place only box and click **OK**.

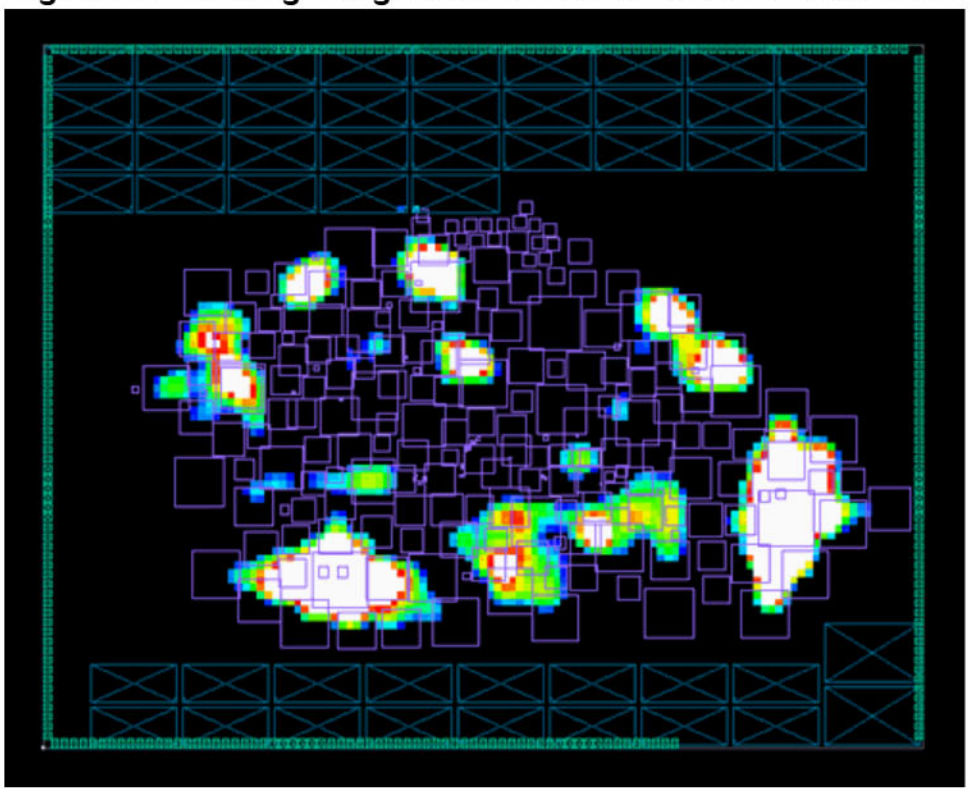
    At the end of the operation, a new physical view appears. Notice the rectangular shape of the design and the more loosely placed blocks (Figure 4-2).

Figure 4-2. Modified Placement of the Design



12. Click the **Routing Congestion Map** icon to view the effect on the routing congestion (Figure 4-3).

Figure 4-3. Routing Congestion View After Modified Placement

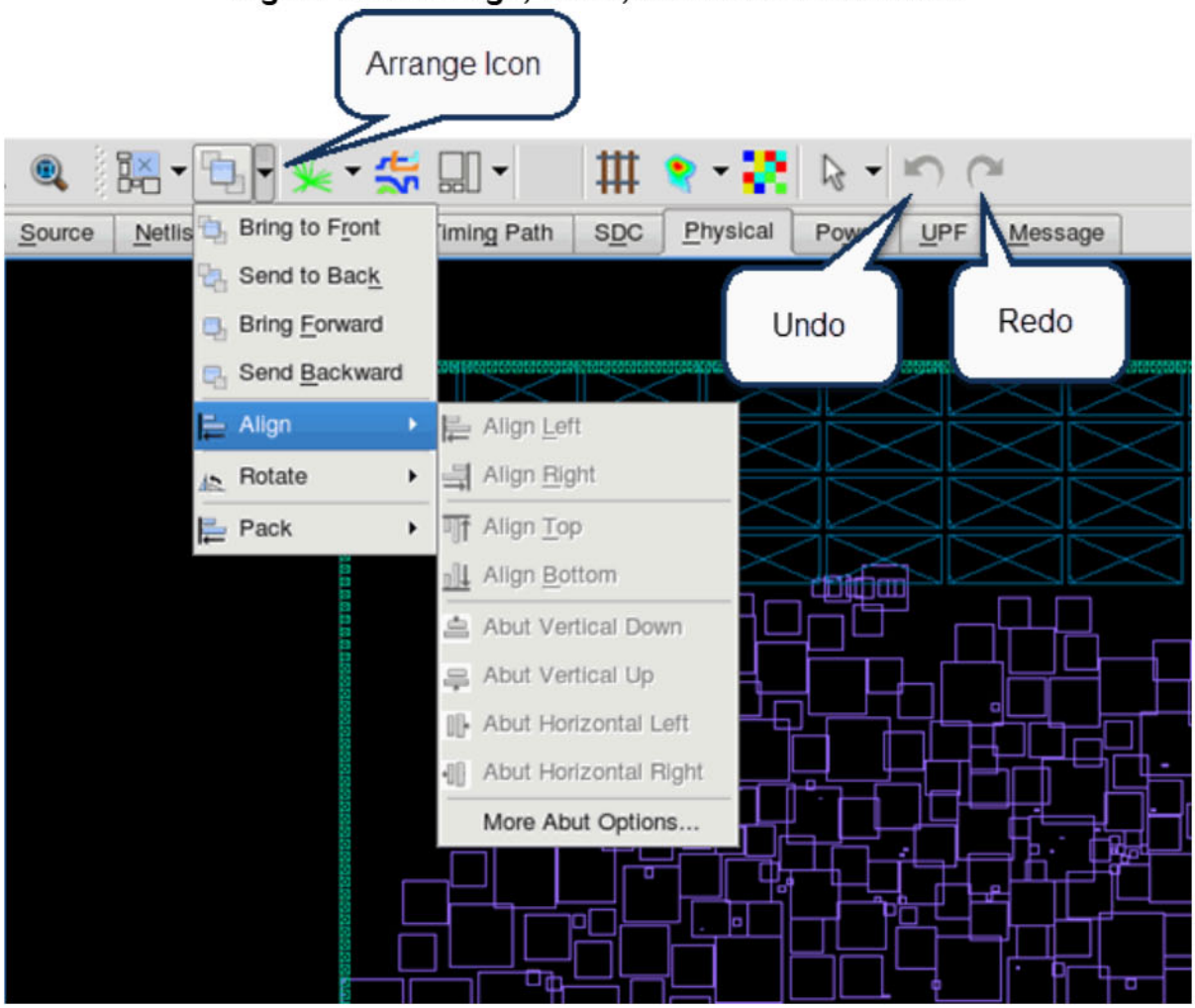


13. (optional) You can use commands available through the **Arrange** icon to perform minor interactive refinements of the macro placement in the floorplan, such as abut, rotate, and move commands (Figure 4-4).

Note

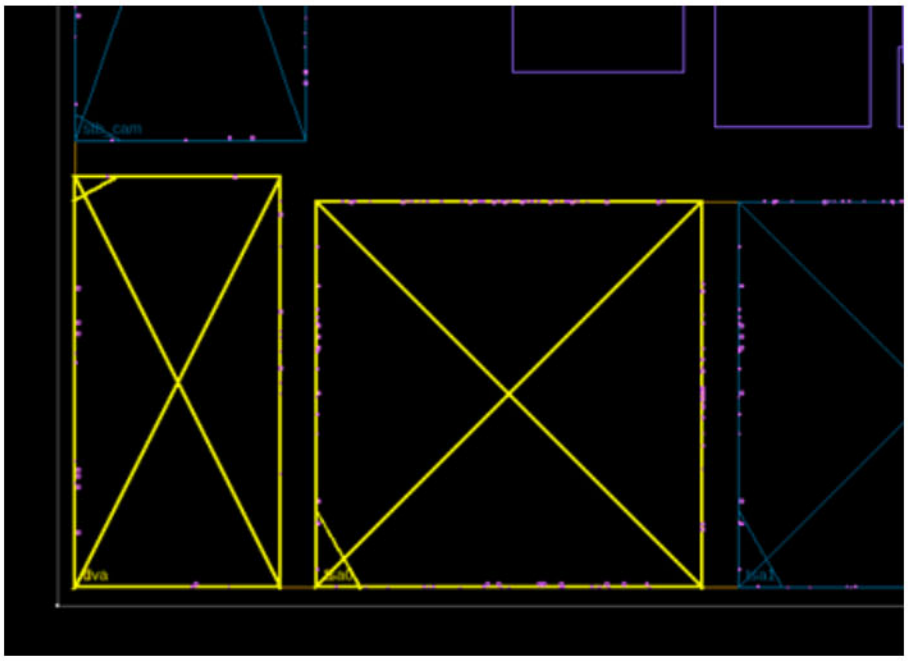
To revert your actions on any manual movement of the macros, you can use the Undo and Redo buttons.

Figure 4-4. Arrange, Undo, and Redo Commands

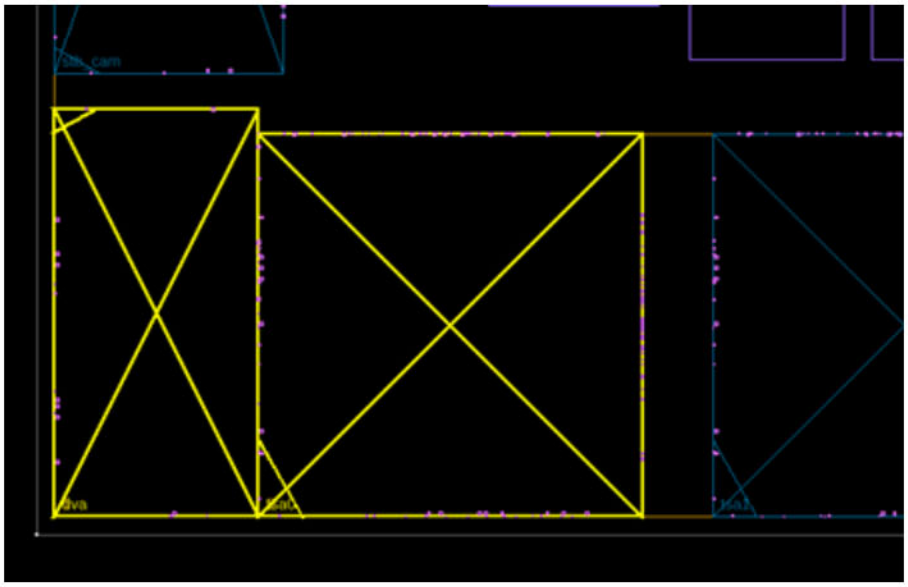


The following steps demonstrate some different ways you can arrange macros.

a. **Abut Macros Horizontally** — Select two macros (for example, as shown in the following figure), then click the **Arrange** icon and select **Align > Abut Horizontal Left**.



The following figure shows the macros after they have been abutted with each other horizontally to the left.



You can use the get_attribute command to verify the movement of the macros and to detect the shift in coordinates:

```
get_attribute -class cell [get_selection] origin
```

The original position of the macros:

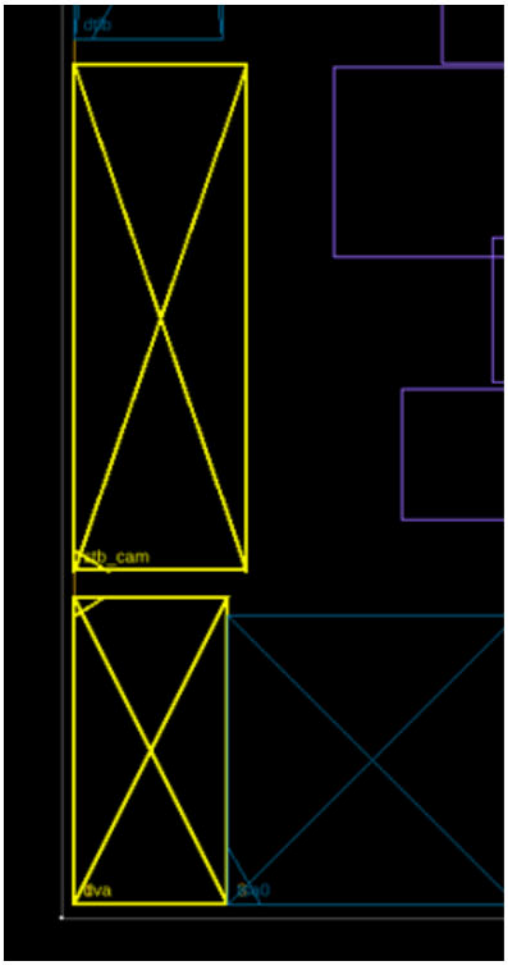
```
73.000000 5.000000 5.000000 5.000000
```

Updated position of the macros:

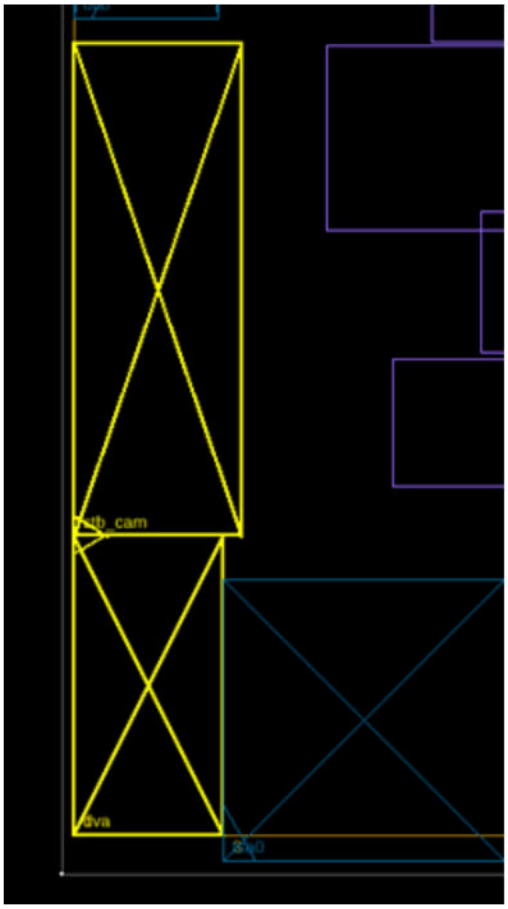
```
63.000000 5.000000 5.000000 5.000000
```

To abut macros horizontally right, click the **Arrange** icon and select **Align > Abut Horizontal Right**.

b. **Abut Macros Vertically** — Select two macros (as shown in the following figure), then click the **Arrange** icon and select **Align > Abut Vertical Up**.



The following figure shows the macros after they have been abutted with each other vertically upward.



Use the get_attribute command to verify the movement of the macros:

```
get_attribute -class cell [get_selection] origin
```

Original position:

```
659.089966 502.399963 758.089966 64.399948
```

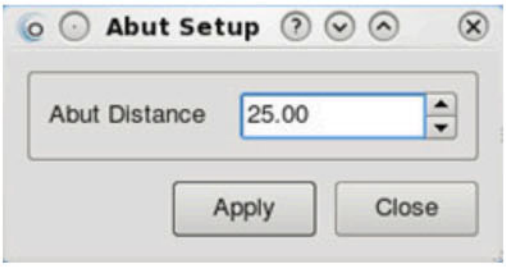
Updated position:

```
659.085022 502.399994 758.085022 74.400002
```

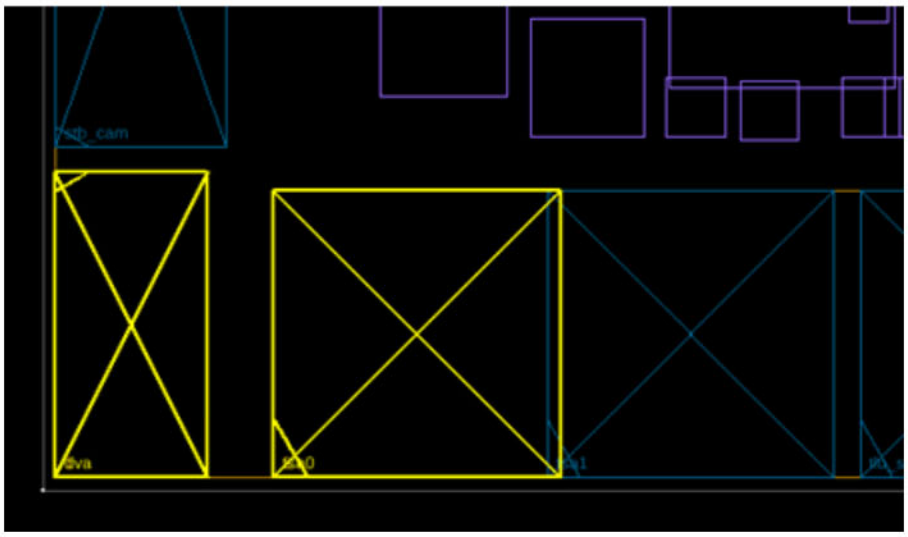
To abut macros vertically downward, click the **Arrange** icon and select **Align > Abut Vertical Down**.

c. **Abut Macros to a Specified Distance** — Click the **Arrange** icon and select **Align > More Abut Options**.

The following dialog box appears:



Enter a distance (in this case, 25u) and click the **Apply** button. As a result, when you perform any of the abut operations, the tool will apply the specified distance. As an example, consider the macros in the following figure:



Select the two macros, then click the **Arrange** icon and select **Align > Abut Horizontal Left**.

As a result, the macros are abutted with an intervening space of 25u. Check that the specified distance has been used using the get_attribute command:

```
get_attribute -class cell [get_selection] origin
```

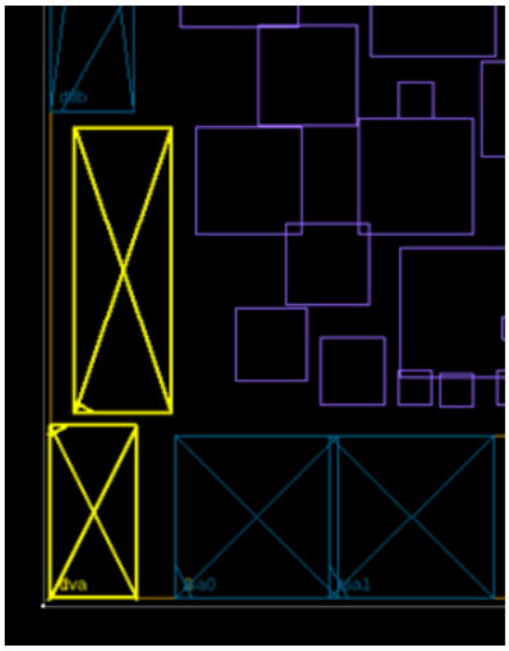
Original position:

```
63.000000 5.000000 5.000000 5.000000
```

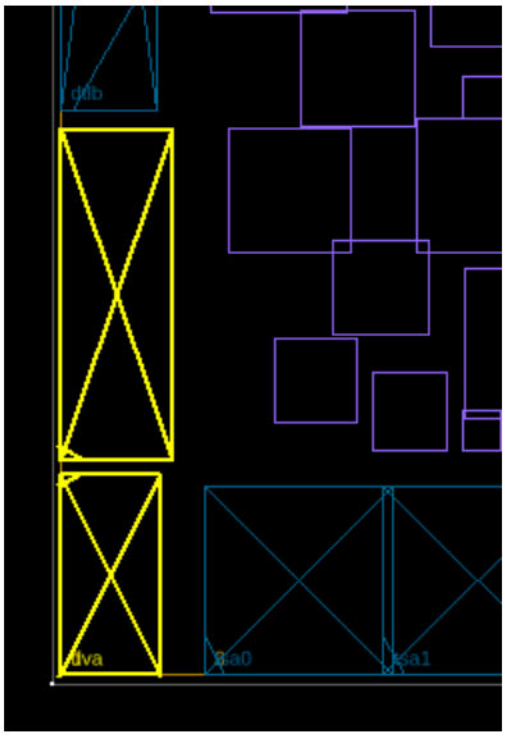
Updated position:

```
88.000000 5.000000 5.000000 5.000000
```

a. **Align a Set Macros Relative to Each Other** — Select two (or more) macros (as shown in the following figure), then click the **Arrange** icon and select **Align > Align Left**.



The following figure shows the macros after they have been aligned with the leftmost macro:



Use the get_attribute command to verify the coordinates of the macros.

get_attribute -class cell [get_selection] origin

Original position:

```
21.365000 129.565002 5.000000 5.000000
```

Updated position:

```
5.000000 129.565002 5.000000 5.000000
```

14. (optional) If you plan to continue with the next procedure, do not exit from the design session.

Related Topics

Constraining a Floorplan with User-Defined Blockages and Regions

Creating a Rectilinear Floorplan

Constraining a Floorplan with User-Defined Blockages and Regions

To add physical constraints to the floorplan, one method is to draw fences and regions. Use fences to confine hierarchical modules by keeping them out of a certain region. Use regions to group blocks together in a designated area.

Physical constraints, called blockages, can be added to the floorplan to prevent placements of cells in specific areas. You can use the create_blockage command or the Create Blockage selection in the GUI to create or modify blockages.

In the DEF layout, a TYPE can be specified for a REGION. By default, the create_region command assigns a TYPE of DEFAULT. All instances assigned to the region are placed inside the region boundaries. Other cells can also be placed inside the region.

For a REGION TYPE of FENCE, all assigned instances must be exclusively placed inside the region boundaries. No other instances are allowed inside the FENCE region.

For TYPE of GUIDE all instances should be placed inside the region; however, it is a preference, not a hard constraint. Other constraints, such as the wire length and timing, can override this preference.

Prerequisites

- A generated design database (odb).

Procedure

1. If you are continuing from "Generating an Initial Floorplan" on page 47, go to the next step. Otherwise, do the following:

   a. Invoke the Oasys-RTL GUI:

      ```
      $OASYS_HOME/bin/oasys -gui -log output/logs/floorplan.log
      ```

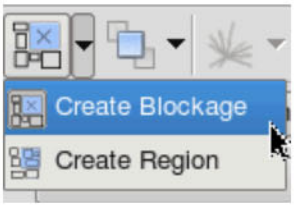
   b. Open the synthesized and optimized Oasys-RTL database generated in module 2:

      read_db output/odb/demo_chip.oasys_final.odb

2. Delete the physical data by entering the following command at the Oasys-RTL prompt:

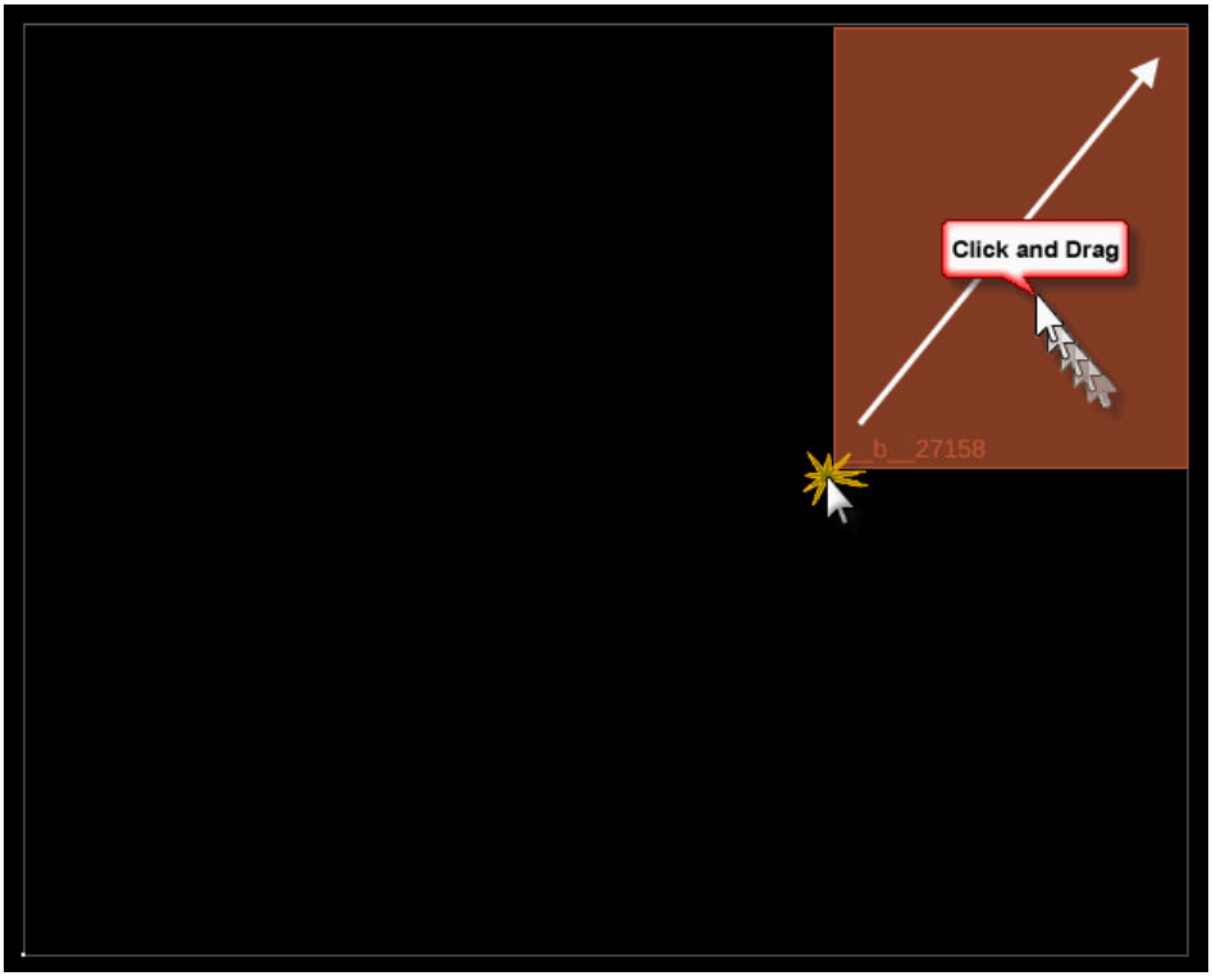
   delete_physical -all

3. Open the **Core Area Editing** form by selecting **Floorplan > Edit Core Area**.
4. In the **Core Area Editing** form, select the Core Utilization and Aspect Ratio box and enter the following values:
   - Utilization = 40
   - Aspect Ratio (Height/Width) = 0.8
   - Delete all physical: enabled
5. Click **OK** to create a new floorplan outline for the design.
6. Click **Yes** if a confirmation dialog appears.
7. Select the **Physical** tab.
8. Click the arrow to the right of the create menu to expand the Create list.



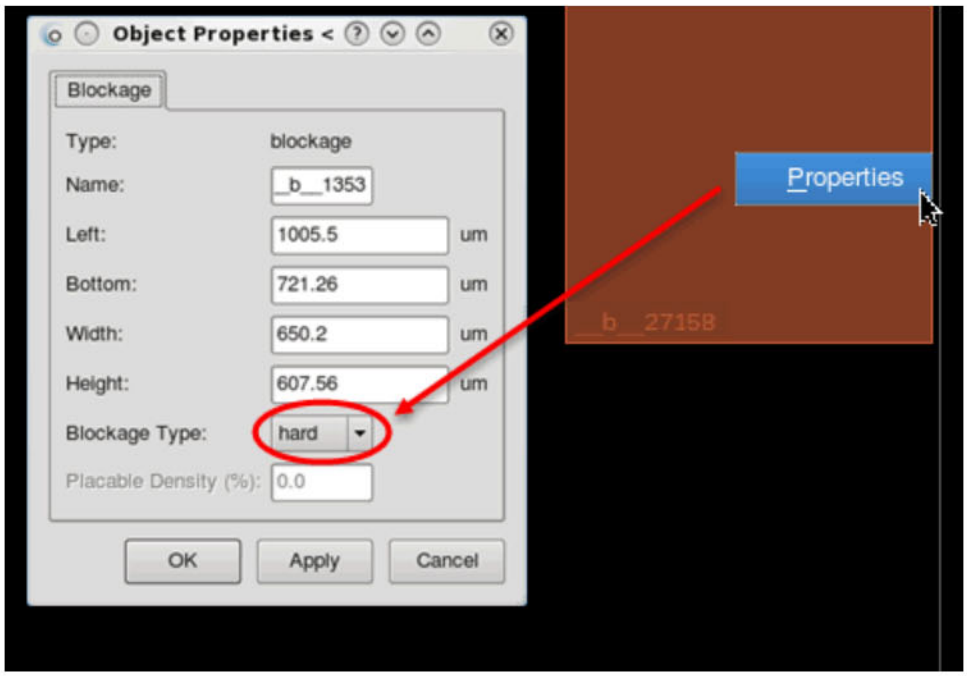
9. Select the **Create Blockage** ( ) from the Create list. The cursor turns to a "+", enabling the drawing function.
10. Draw a rectangle in the upper right corner of the floorplan area.

To draw a rectangle, click inside the floorplan area and drag the mouse to the upper right corner of the floorplan outline.



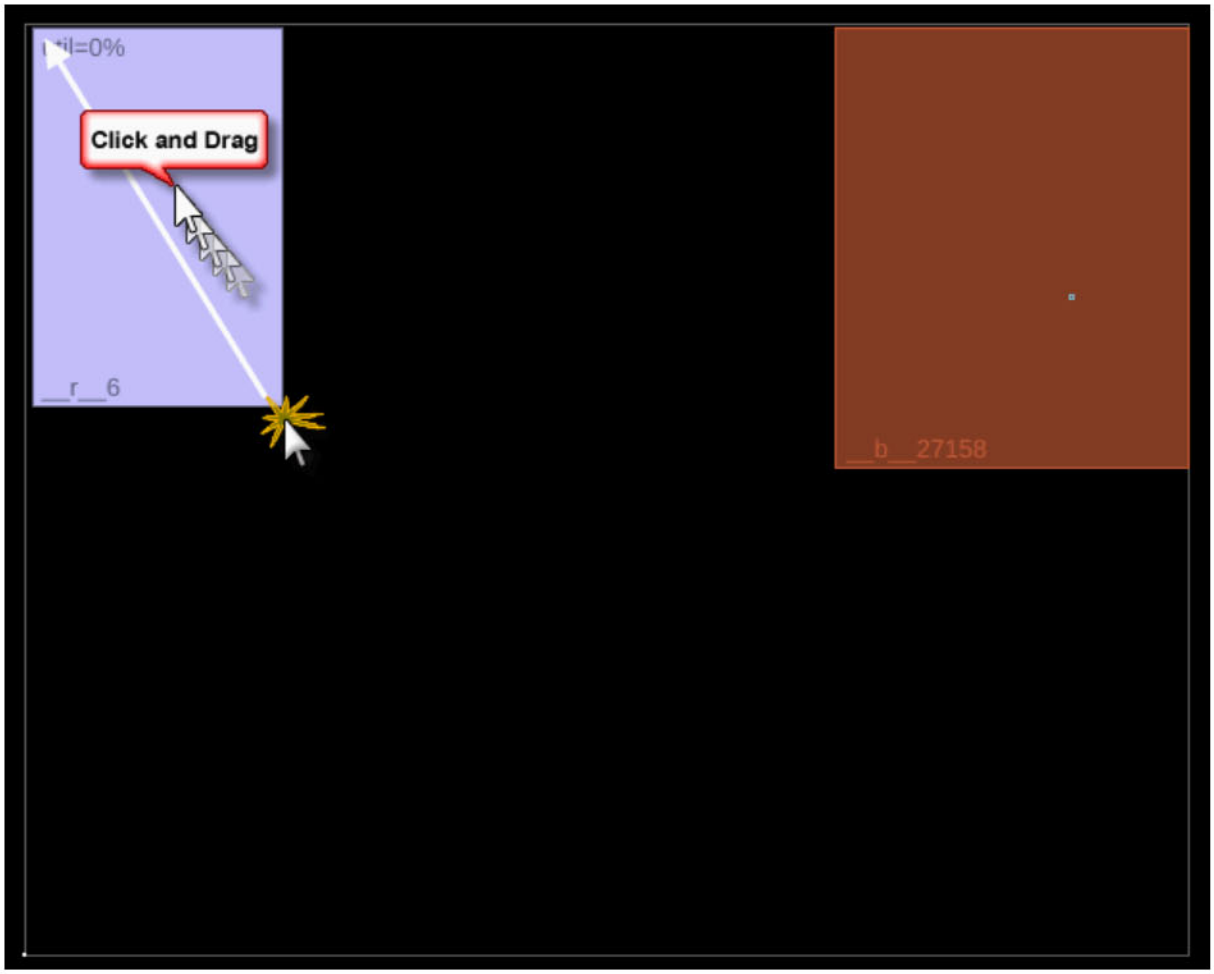
Note

By default the Oasys-RTL tool creates only hard blockages, but if you want to create soft and partial blockages, right-click in the rectangle and select Properties. In the Object Properties dialog box, modify the Blockage Type setting.

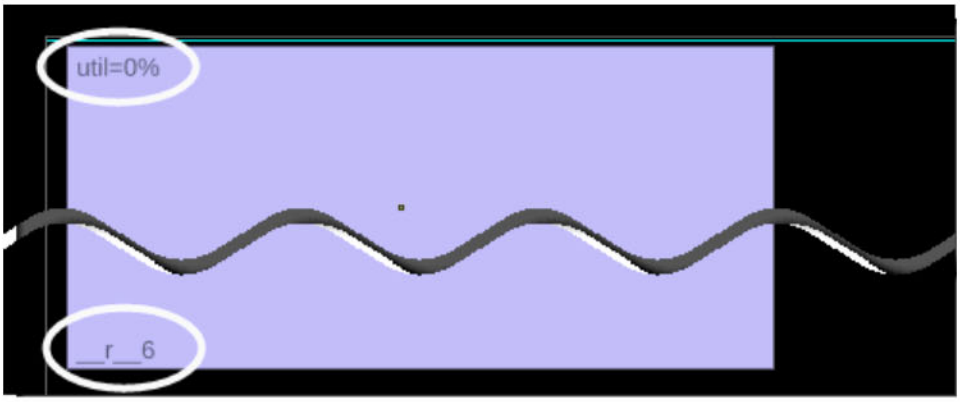


11. Select **Create Region** ( ) tool from the Create list.

12. Similar to creating a blockage, create a region in the upper left corner of the floorplan area.



Notice the text in the upper-left corner of the region. This gives the current area utilization for the region. Find the name of the region in the bottom-left corner.

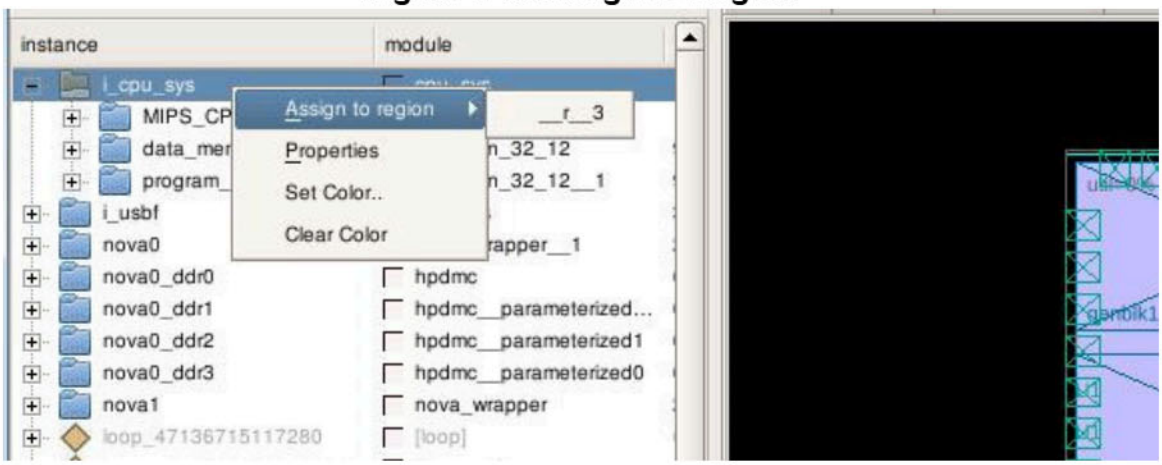


13. Assign the i_cpu_sys block to the region (__r__*n*) as follows:

    a. In the **Instance** list, select the i_cpu_sys block and right-click to open the pull-down menu.

    b. From the menu, select **Assign to region**. A list of regions appears; in this case there is only one region.

Figure 4-5. Assign to Region

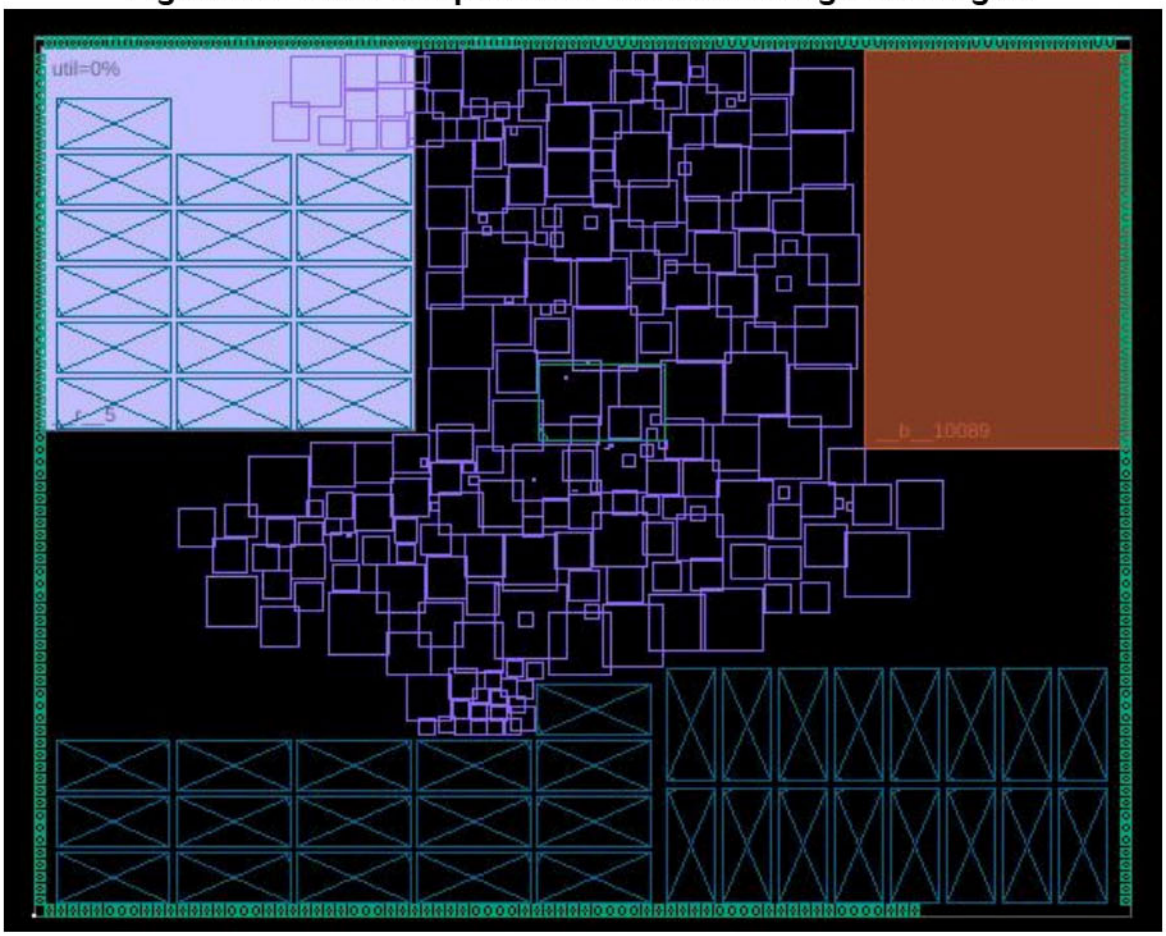


    c. Select the region "__r__*n*". Notice that the utilization percentage changes (util=%).

14. To complete the placement, select **Flow > Optimize** and select the Place only box in the Optimize dialog box. Click **OK**. This step takes some time to execute.

15. The Physical tab shows the new floorplan with the drawn blockage and region. To verify that the i_cpu_sys block was placed inside __r__*n* region, select it from the **Instance** list.

Figure 4-6. New Floorplan With Drawn Blockage and Region



16. (optional) If you plan to continue with the next procedure, do not exit from the design session.

Related Topics

Generating an Initial Floorplan

Constraining a Floorplan by Reading a DEF File

Constraining a Floorplan by Reading a DEF File

To add physical constraints to the floorplan, another method is to read the fence and regions from a DEF file.

Use fences to confine hierarchical modules by keeping them out of a certain region. Use regions to group blocks together in a designated area.

Prerequisites

- A generated design database (odb).

Procedure

1. If you are continuing from "Constraining a Floorplan with User-Defined Blockages and Regions" on page 60, go to the next step. Otherwise, do the following:

   a. Invoke the Oasys-RTL GUI:

      ```
      $OASYS_HOME/bin/oasys -gui -log output/logs/floorplan.log
      ```

   b. Open the synthesized and optimized Oasys-RTL database generated in Chapter 2, "Design Synthesis":

      read_db output/odb/demo_chip.oasys_final.odb

2. Delete the physical data by entering the following command at the Oasys-RTL prompt in the GUI:

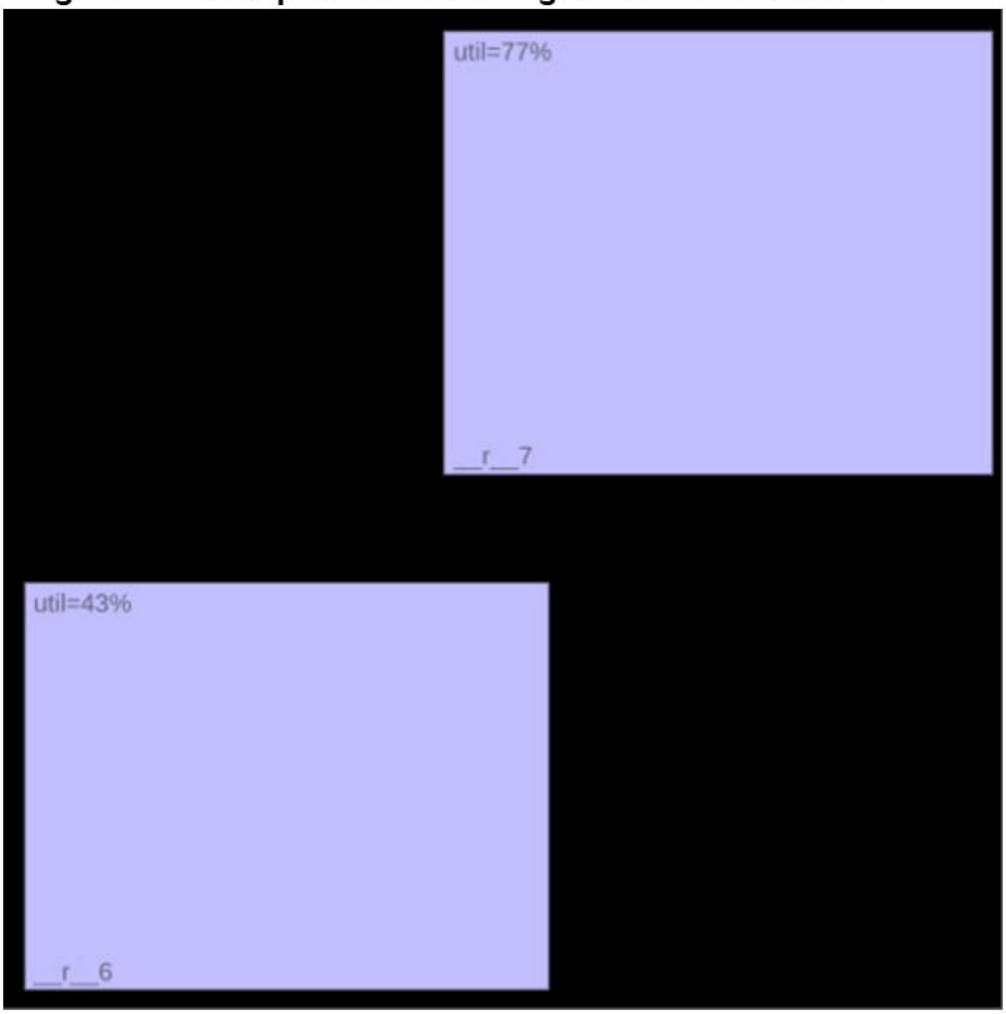
   delete_physical -all

3. Load the DEF file for placement by entering the following command at the Oasys-RTL prompt:

   read_def constraints/define_regions.def

   This loads the floorplan with two regions defined in the DEF file for blocks i_cpu_sys and i_usbf (Figure 4-7).

Figure 4-7. Floorplan with Two Regions Defined in the DEF File



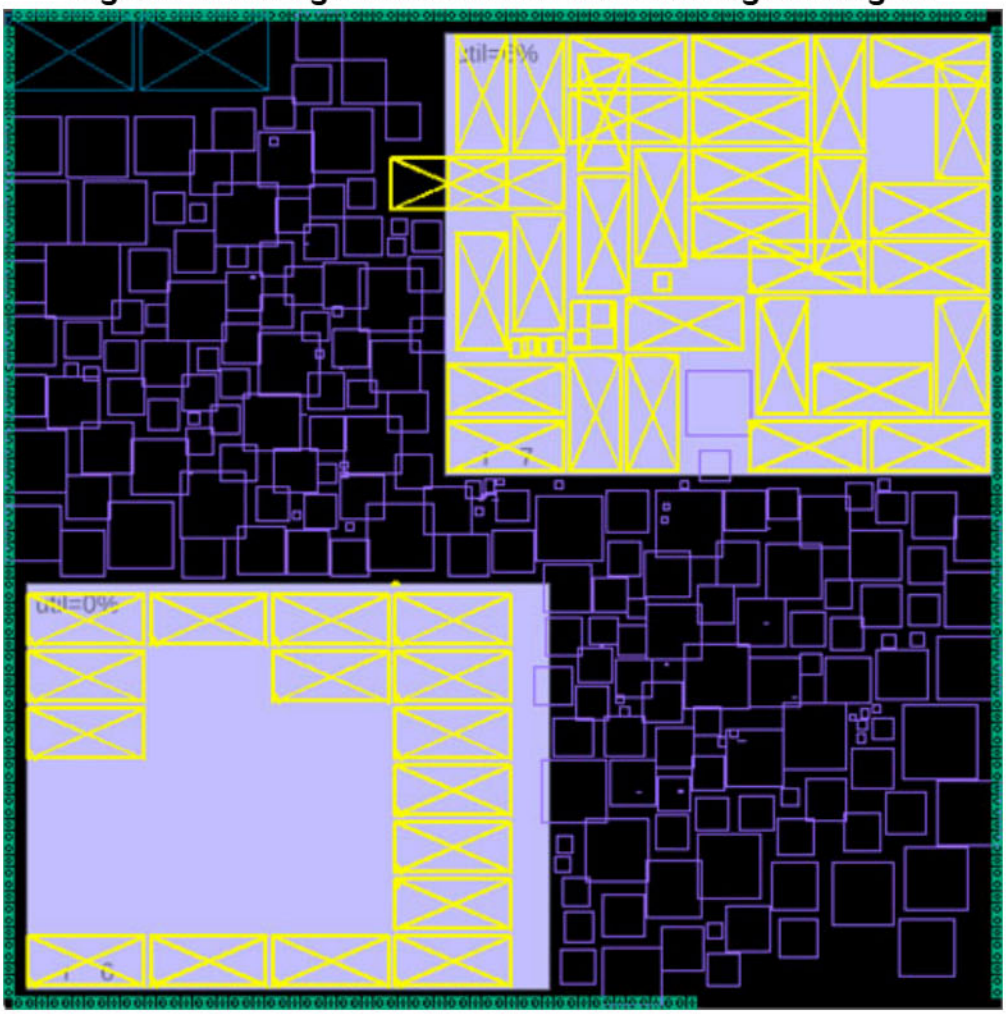
4. To complete the placement, enter the following command at the Oasys-RTL prompt:

   optimize -place

5. Review the placement of the design. Notice that the i_cpu_sys and i_usbf blocks are placed in their assigned regions (Figure 4-8).

   To verify this, multi-select the i_cpu_sys and i_usbf blocks in the **Instance** list by first selecting the i_cpu_sys block, then pressing Ctrl-Left-click and selecting the i_usbf block.

Figure 4-8. Design with Blocks Place in Assigned Region



6. (optional) If you plan to continue with the next procedure, do not exit from the design session.

Related Topics

Constraining a Floorplan with User-Defined Blockages and Regions

Generating an Initial Floorplan

Creating a Rectilinear Floorplan

It is possible to create floorplans with a rectilinear footprints by using a set of coordinates.

For many designs and SoCs, a rectilinear shape is required to optimize chip area and space. Oasys-RTL floorplanning supports rectilinear shapes.

Prerequisites

- A generated design database (odb).

Procedure

1. If you are continuing from "Constraining a Floorplan by Reading a DEF File" on page 66, go to the next step. Otherwise, do the following:

   a. Invoke the Oasys-RTL GUI:

      ```
      $OASYS_HOME/bin/oasys -gui -log output/logs/floorplan.log
      ```

   b. Open the synthesized and optimized Oasys-RTL database generated in Chapter 2, "Design Synthesis":

      read_db output/odb/demo_chip.<dft>_post_fix.odb

2. Load the rectilinear floorplan by entering the following at the Oasys-RTL prompt:

   source scripts/rectilinear.tcl

   This removes any existing physical design information and creates an L-shaped floorplan area (Figure 4-9).

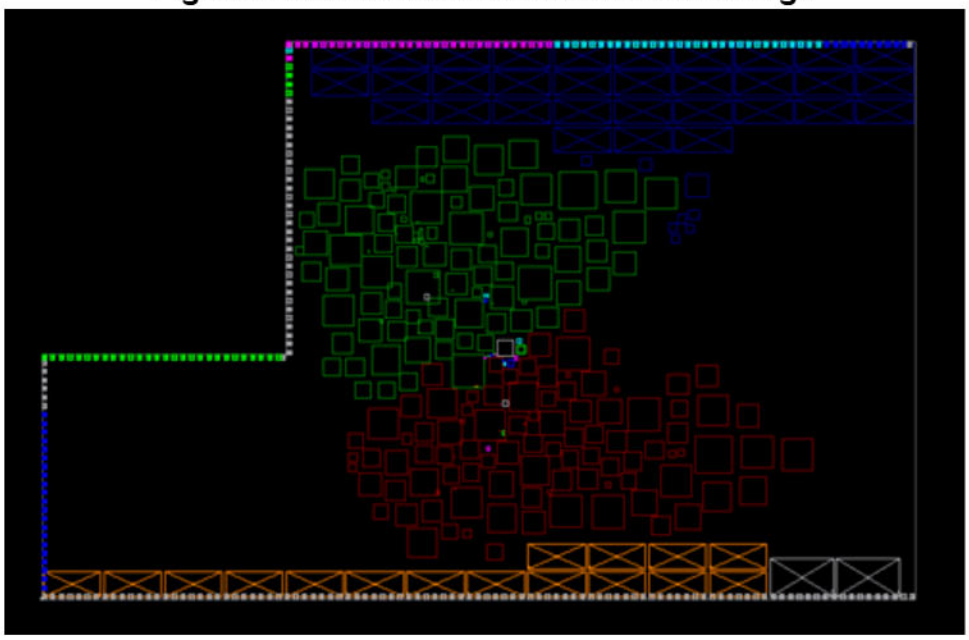
Figure 4-9. L-Shape Floorplan Area

3. To complete the placement, select **Flow > Optimize** and select the Place only box in the Optimize dialog box. Click **OK**.

4. Review the placement of the design.

   To display hierarchical view as shown in Figure 4-10, select **View > Hierarchical View** from the main menu. The colors correspond to the hierarchy grouping in the RTL.

Figure 4-10. Hierarchical View of the Design



5. Exit the Oasys-RTL tool.

Related Topics

Constraining a Floorplan by Reading a DEF File

Generating an Initial Floorplan