

PlanAhead Software Tutorial

Leveraging Design Preservation for Predictable Results

UG747 (v 13.1) March 1, 2011



The information disclosed to you hereunder (the "Information") is provided "AS-IS" with no warranty of any kind, express or implied. Xilinx does not assume any liability arising from your use of the Information. You are responsible for obtaining any rights you may require for your use of this Information. Xilinx reserves the right to make changes, at any time, to the Information without notice and at its sole discretion. Xilinx assumes no obligation to correct any errors contained in the Information or to advise you of any corrections or updates. Xilinx expressly disclaims any liability in connection with technical support or assistance that may be provided to you in connection with the Information. XILINX MAKES NO OTHER WARRANTIES, WHETHER EXPRESS, IMPLIED, OR STATUTORY, REGARDING THE INFORMATION, INCLUDING ANY WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NONINFRINGEMENT OF THIRD-PARTY RIGHTS.

© Copyright 2011 Xilinx, Inc. XILINX, the Xilinx logo, Virtex, Spartan, ISE, and other designated brands included herein are trademarks of Xilinx in the United States and other countries. All other trademarks are the property of their respective owners.

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
03/01/2011	13.1	Update for the 13.1 release.

Table of Contents

Revision History	2
PlanAhead Software Tutorial: Leveraging Design Preservation for Predictable Results	
Introduction	5
Tutorial Objectives.....	5
Getting Started.....	5
Tutorial Steps.....	6
Step 1: Opening an Existing PlanAhead RTL Project and Elaborating the RTL Design.....	7
Step 2: Setting Partitions and Drawing Pblocks	9
Step 3: Synthesizing and Implementing the Design.....	13
Step 4: Promoting Successfully Implemented Partitions.....	16
Step 5: Modifying the RTL for Top	21
Step 6: Rerunning Synthesis and Implementation While Importing	22
Conclusion.....	23
Appendix A: Additional Resources	
Xilinx Resources	25
PlanAhead Documentation.....	25

PlanAhead Software Tutorial: Leveraging Design Preservation for Predictable Results

Introduction

This tutorial provides an overview of the Design Preservation flow, in which you will:

- Define Partitions and Pblocks on an elaborated Register Transfer Level (RTL) design
- Synthesize using Xilinx® Synthesis Technology (XST) Incremental synthesis
- Implement the partitioned design
- Promote successful implementation results
- Update the top-level partition
- Rerun synthesis and implementation on the modified top level while importing the unchanged Partitions

Many of the PlanAhead™ software analysis features are covered in more detail in other tutorials, and not every command or command option is covered.

Tutorial Objectives

The objective of this tutorial is to familiarize you with the partitions and the Design Preservation flow using the PlanAhead software.

Getting Started

Software Requirements

The PlanAhead software is installed with ISE® Design Suite software. Before starting the tutorial, be sure that the PlanAhead software is operational, and that the tutorial design data is installed.

For installation instructions and information, see the *ISE Design Suite: Installation and Licensing Guide (UG798)* cited in [Appendix A, Additional Resources](#).

Hardware Requirements

Xilinx recommends a minimum of 2 GB of RAM when using the PlanAhead software on larger devices. For this tutorial, a smaller xc6vlx75t design is used, and the number of designs open at one time is limited. Although 1 GB is sufficient, it can impact performance.

Tutorial Design Description

The sample design used in this tutorial has two sets of synthesis results. The first was created using a standard top-down synthesis flow to be used in a flat implementation. The second set was created using an incremental synthesis flow to be used in the Design Preservation flow, and has separate netlists for module instances that will be partitioned. The design used throughout this tutorial contains:

- A RISC processor
- FFTs
- Gigabit transceivers
- Two USB port modules (to be partitioned)
- An xc6vlx75tff784 device

A small design is used to allow the tutorial to be run with minimal hardware requirements and to enable timely completion, as well as to minimize the data size.

Locating Tutorial Design Files

This tutorial uses the design data that is included with the example projects in the PlanAhead software. It is also available on the Xilinx website.

1. Download the `PlanAhead_Tutorial.zip` file from either:
 - The example projects area in the PlanAhead software installation:
<ISE_install_area>/PlanAhead/testcases/
 - The Xilinx website: http://www.xilinx.com/support/documentation/dt_planahead_planahead13-1_tutorials.htm
2. Extract the zip file contents into any write-accessible location.

The unzipped `PlanAhead_Tutorial` data directory is referred to in this tutorial as <Extract_Dir>.

The tutorial sample design data is modified while performing this tutorial. A new copy of the original `PlanAhead_Tutorial` data is required each time you run the tutorial.

Tutorial Steps

This tutorial consists of the following steps:

[Step 1: Opening an Existing PlanAhead RTL Project and Elaborating the RTL Design](#)

[Step 2: Setting Partitions and Drawing Pblocks](#)

[Step 3: Synthesizing and Implementing the Design](#)

[Step 4: Promoting Successfully Implemented Partitions](#)

[Step 5: Modifying the RTL for Top](#)

[Step 6: Rerunning Synthesis and Implementation While Importing](#)

Step 1: Opening an Existing PlanAhead RTL Project and Elaborating the RTL Design

This tutorial takes advantage of preexisting PlanAhead software projects to simplify the steps and focus on the Design Preservation aspects of the tutorial. For a real design the New Project Wizard would be used to create either an RTL or netlist based project. PlanAhead software, version 13.1 or newer, supports for RTL projects with partitions.

Opening an Existing PlanAhead RTL Project

1. Open the PlanAhead software.
 - On Windows, double-click the Xilinx® PlanAhead 13 Desktop icon, or select **Start > Programs > Xilinx ISE Design Suite 13.1 > PlanAhead > PlanAhead**.
 - On Linux, go to `<ISE_install_dir>/PlanAhead_Tutorial/Tutorial_Created_Data` directory and type **planAhead**.
2. From the Getting Started page, click **Open Project**.
3. Browse to the `<Extract_Dir>` and open the project file located at `./Projects/project_DP_RTL/project_DP_RTL.ppr`.

When the project opens, the **Project Manager** view is visible. You can browse the sources for the design in the **Sources** window and note the various VHDL and Verilog files along with a User Constraints File (UCF) called `top_full.ucf`. This UCF already has timing constraints and I/O pin locations.

Elaborating an RTL Design

The **RTL Design** view must be used to define partitions in an RTL project. When the **RTL Design** view is opened the RTL code is elaborated, and the design hierarchy is displayed. This is a pre-synthesized view of the design, and can be used to define partitions and create constraints.

1. Select **Flow > RTL Design**, or click **RTL Design** in the Flow Navigator along the left hand side of the PlanAhead software GUI. See [Figure 1-1](#).

A message about Critical Warnings in the design appears.

2. Click **OK** to close the pop-up window.

Note: Critical Warnings - When the RTL Design view opens, it parses the UCF file. Because the RTL design view is a pre-synthesized version of the design, there are instances with constraints in the UCF file that do not exist in the elaborated design (such as I/O buffers). Because of this, the message is expected and can be safely ignored.

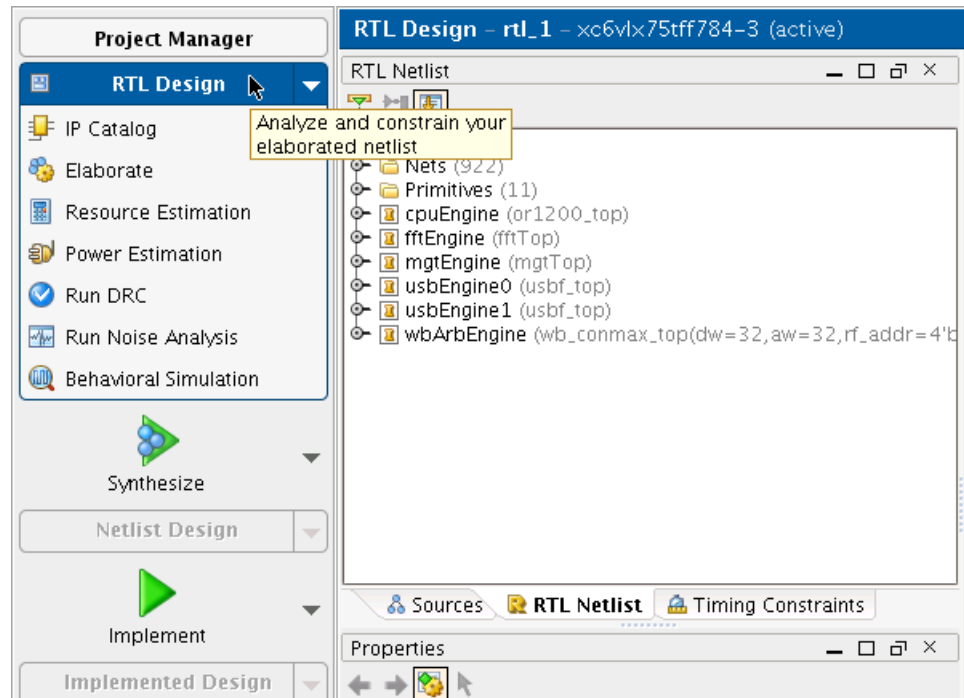


Figure 1-1: Opening the RTL Design View



Step 2: Setting Partitions and Drawing Pblocks

Based on the results of the *PlanAhead Tutorial: Design Analysis and Floorplanning for Performance (UG676)* cited in [Appendix A, Additional Resources](#), the `usbEngine` instances have been identified as timing-critical modules, and it would be advantageous to preserve successful implementation results of these instances. However, this fact alone does not make these good candidates for partitions. These are good choices for partitions because they are logically isolated from the rest of the design, and have reasonable interface timing (registered inputs and outputs). You can use DRCs to help identify whether or not a module is a good choice for partitions. For more information on how to choose good module instances for partitioning, refer to the *Hierarchical Design Methodology Guide (UG748)* cited in [Appendix A, Additional Resources](#).

Partitioned instances can be floorplanned just like any other instance, and creating Pblock (AREA_GROUP) constraints can help achieve timing closure and improve runtime. The UCF provided with this tutorial constrains the I/O logic of `usbEngine` along the left side of the device, and the steps below walk you through creating appropriate Pblock constraints for the two `usbEngine` instances.

Setting Partitions for the Two `usbEngine` Instances

1. From the **RTL Design** view, use the **Netlist** tab to select the two `usbEngine` instances.
2. Right-click and select **Set Partition** (Figure 1-2).

Once the modules have been set as partitions, the icon associated with the instances changes from  to .

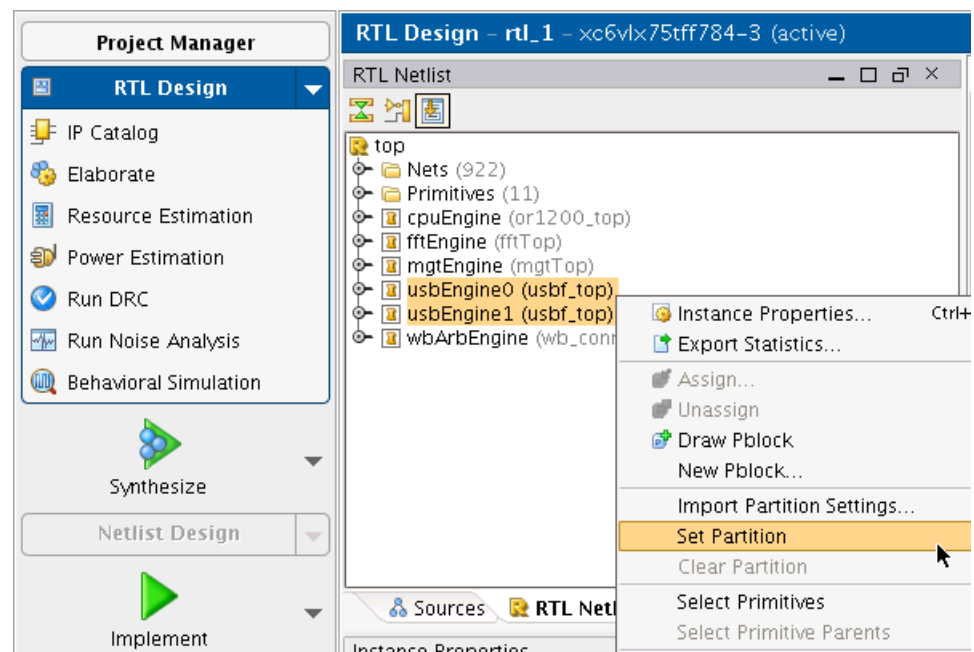



Figure 1-2: Setting Partitions on `usbEngine` Instances

Drawing Pblocks for the Two usbEngine Instances

This step does not affect synthesis results, and could be done post synthesis in the Netlist Design view if desired. There are actually some advantageous to this such as resource estimation to help size the Pblocks. For this tutorial, you will draw Pblocks on the pre-synthesized design from the RTL Design view just to demonstrate how it can be done.

1. From the Netlist window, select and right-click **usbEngine1**, and select **Draw Pblock** (Figure 1-3).

Note: You can use the **Draw Pblock** button  from the **Device** view toolbar.

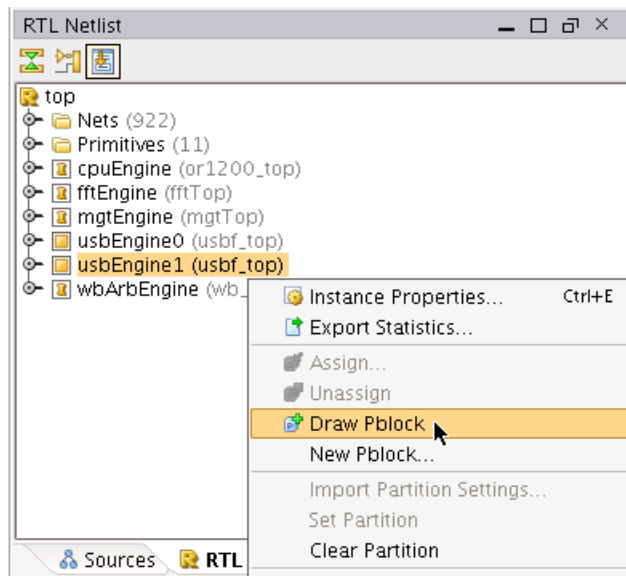


Figure 1-3: Selecting the Draw Pblock Tool

2. With the Draw Pblock tool active, move the cursor to the **Device** window.
3. Left click on the top-left corner of the device where the CLBs start, and without letting go of the left mouse button, drag to create a rectangle covering most of the top-left quadrant of the device (Figure 1-4).
4. In the **New Pblock** dialog box, verify that the SLICE and RAMB36 grids are selected, and deselect other resources that are not needed (Figure 1-4).

Verify that number of available RAMB36 (shown in parenthesis) is 36. If the rectangle does not fully cover the region shown in Figure 1-4, this number might be less, and the design will fail to place.

5. Click **OK**.

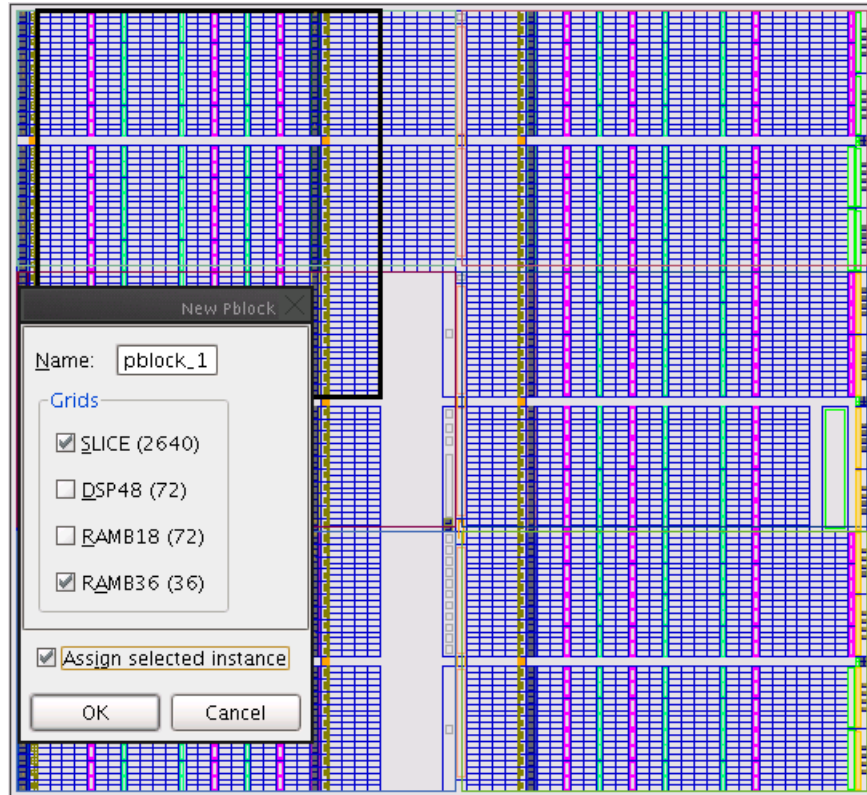


Figure 1-4: Pblock Rectangle for usbEngine1

The most important thing to note here is the number of available RAMB36 resources. If the Pblock rectangle does not completely cover the RAMB36 resources as shown in [Figure 1-5](#), the number of resources could be less than the required 36. If this is the case, adjust the size of the Pblock rectangle by selecting it and resizing.

- Repeat Steps 1 to 5 for usbEngine0 on the bottom-left quadrant.

The completed floorplan should look like [Figure 1-5](#).

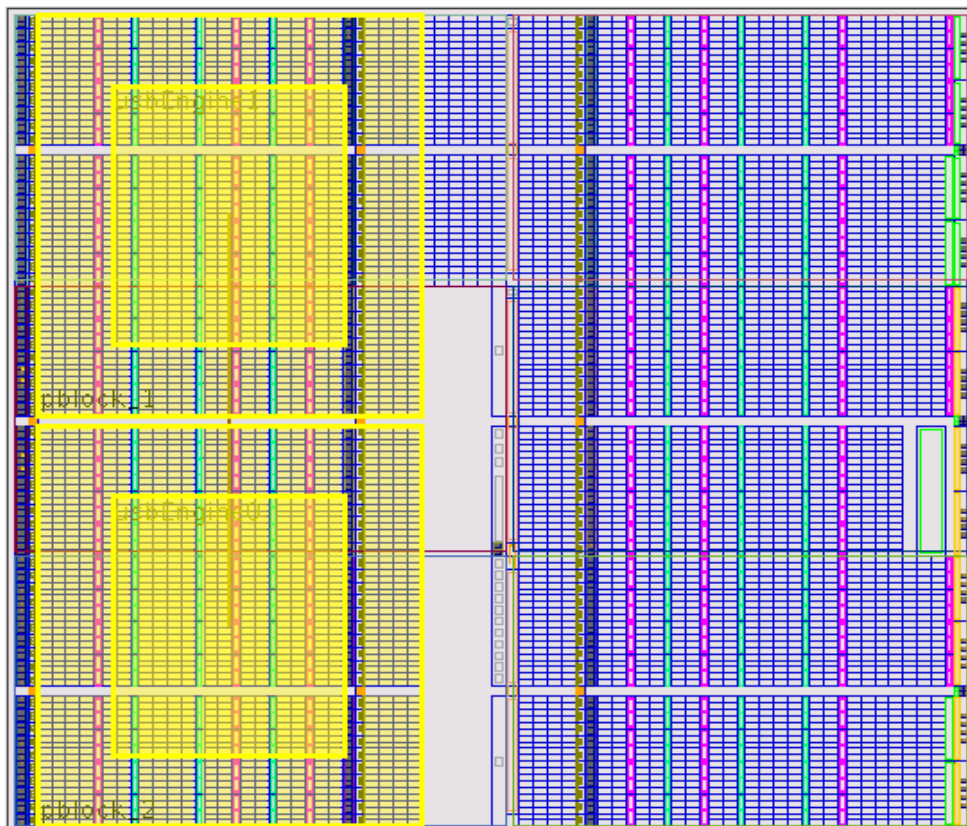


Figure 1-5: Completed Floorplan for Pblocks usbEngine0 and usbEngine1

Step 3: Synthesizing and Implementing the Design

Now that the design has partitions defined and all necessary constraints created, synthesis and implementation can be run. Because we defined partitions on the elaborated Hardware Description Language (HDL) design, XST picks up the partitions and runs its incremental flow. This creates an individual NGC file for each partition that is defined. These netlists are uniquely named to allow for multiple instances of a module to be synthesized with different parameters. In this case, synthesis generates the following NGC files (located in `<project_name>.runs/synth_1` directory):

- `top.ngc`
- `usbEngine0#usbf_top.ngc`
- `usbEngine1#usbf_top.ngc`

If bottom-up synthesis or third party incremental synthesis flow is desired, then synthesis can be run outside of the PlanAhead software, and a PlanAhead netlist project would be used instead of the RTL project shown in this tutorial.

Running Synthesis

1. To launch synthesis, click the big **Synthesize** button in the Navigator pane. To set specific XST options or check the settings on the partitions, the pull-down menu on the right side of the Synthesize button can be used. Simply click the down arrow and select **Synthesis Settings**. For this design the defaults are fine, so just click the big green synthesis button as shown in [Figure 1-6](#).

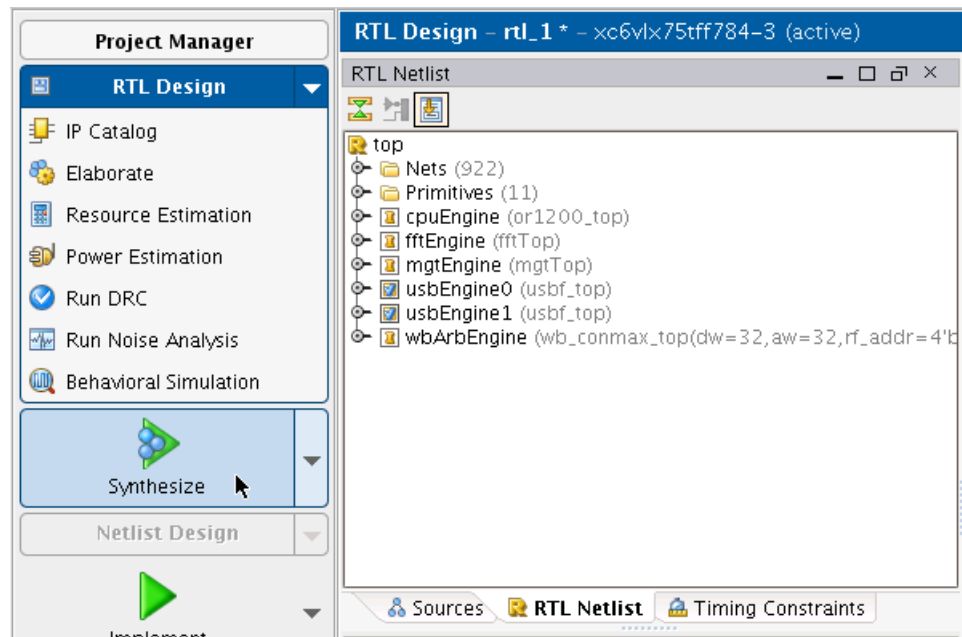


Figure 1-6: Launch XST Incremental Synthesis

2. If you have not already saved the design, click on **Save** in the popup window to save the partition and floorplan changes prior to launching synthesis.

Running DRC on Partitions

Design Rule Checks (DRC) can be run on the RTL Design, but there are limitations to what the tools can check at that stage. However, DRC should definitely be run prior to launching implementation. To do this, you load the **Netlist Design** view, and then run partition-specific DRCs.

1. Open the Netlist Design view by clicking on **Netlist Design** in the Navigator. This loads the synthesis results and allows for additional DRC checking.
2. In the Navigator under **Netlist Design**, click **Run DRC** (or go to **Tools > Run DRC**).
3. From the **Run DRC** dialog box, unselect all rules except **Partition**, and click **OK** (Figure 1-7).

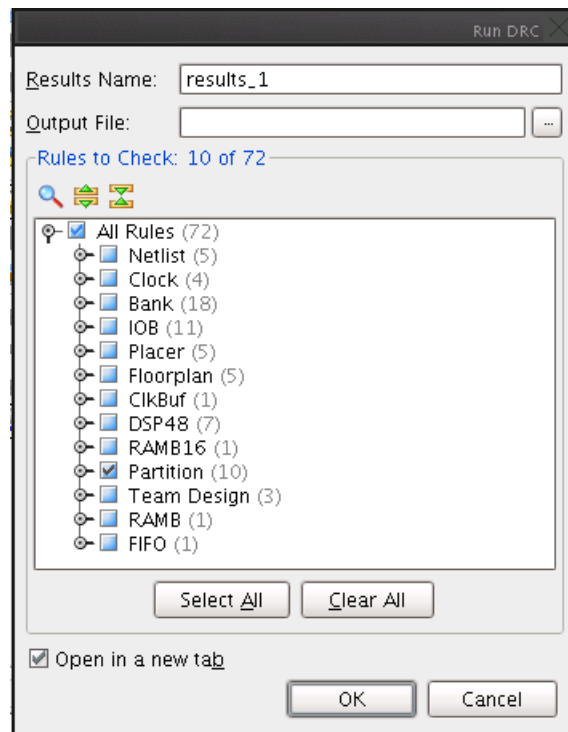


Figure 1-7: Run Partition DRCs

Note: The DRC returns minor Advisory messages. The PlanAhead software can report Advisory, Warning, Error, and Fatal messages for the DRC rules. You can ignore the Advisory messages given for this tutorial design. For an actual design, however, all messages reported by the DRCs should be understood and dealt with properly.

Running Implementation

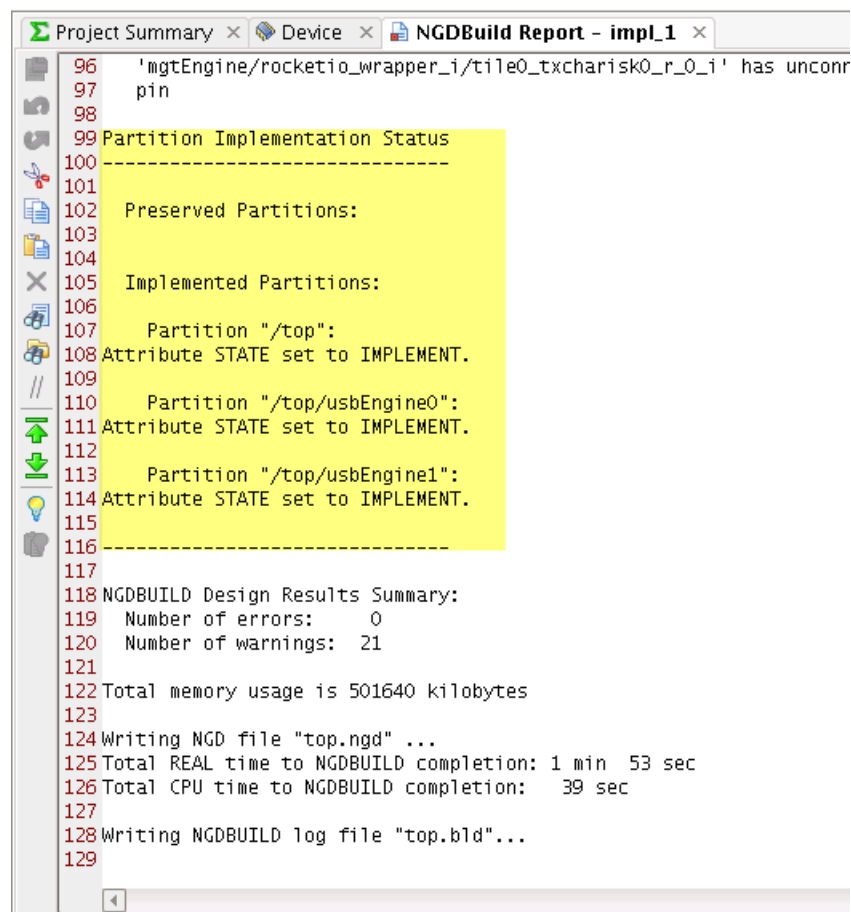
Implementation is now ready to be run. To implement this design with partitions, the only additional steps required were to define the partitions and to run a quick DRC check. This design was created with hierarchy in mind, so the design did not need to be altered to work with partitions. However, the bulk of the work required to make a partition design successful is at the RTL design stage, and not with the synthesis or implementation tools. For recommendations for good hierarchical design, refer to the *Hierarchical Design Methodology Guide (UG748)* cited in [Appendix A, Additional Resources](#).

To remove the implementation runtime from this tutorial, a completed project with synthesis and implementation results can be found at: <Extract_Dir>/Projects/project_DP_RTL_implemented/project_DP_RTL_implemented.ppr

Use the **File > Open Project** command to skip this step and open the completed results. If you skip this step, proceed to [Step 4: Promoting Successfully Implemented Partitions](#).

1. In the Flow Navigator, click **Implement**.
2. Click the **Report** tab to get a list of all the ISE® Implementation report files. As each process finishes, the related report files are available to open.
3. Once NGDBuild finishes, double-click the **NGDBuild Report** to open the report file. Scroll to the bottom of the report file and note the partition information ([Figure 1-8](#)).

This information is provided in every report file (NGDBuild, Map, and PAR) and is an easy way to verify the status of all partitions on a given run.



```
96 'mgtEngine/rocketio_wrapper_i/tile0_txcharisk0_r_0_i' has uncon
97 pin
98
99 Partition Implementation Status
100 -----
101
102 Preserved Partitions:
103
104 Implemented Partitions:
105
106 Partition "/top":
107 Attribute STATE set to IMPLEMENT.
108
109 Partition "/top/usbEngine0":
110 Attribute STATE set to IMPLEMENT.
111
112 Partition "/top/usbEngine1":
113 Attribute STATE set to IMPLEMENT.
114
115 -----
116
117
118 NGDBUILD Design Results Summary:
119 Number of errors: 0
120 Number of warnings: 21
121
122 Total memory usage is 501640 kilobytes
123
124 Writing NGD file "top.ngd" ...
125 Total REAL time to NGDBUILD completion: 1 min 53 sec
126 Total CPU time to NGDBUILD completion: 39 sec
127
128 Writing NGDBUILD log file "top.bld"...
129
```

Figure 1-8: Partition Implementation Status in Report Files

Step 4: Promoting Successfully Implemented Partitions

Once an implementation is successful, the results can be promoted. Promoting results makes a copy of the implementation directory in `<project_name>.promote\X<run_name>` (for example, `project_DP_RTL.promote\Ximpl_1`).

The PlanAhead software keeps track of the latest promoted run and automatically changes the state and import location of any promoted partitions. This can all be managed manually in the PlanAhead software.

Promoting the Successful Implementation Results

1. Once the implementation completes, the Implementation Completed dialog box appears, providing several choices ([Figure 1-9](#)).

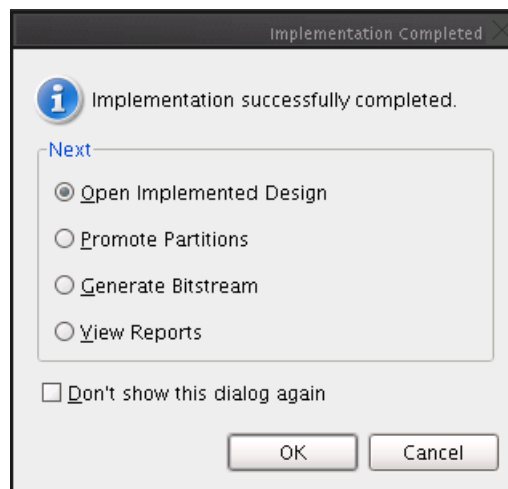


Figure 1-9: Implementation Completed Dialog Box

2. Select **Open Implemented Design** and click **OK**.
Note the other options available. The results could be promoted here using the **Promote Partitions** option, but this tutorial promotes the results another way.
If you did not see the Implementation Completed dialog box, if you opened the completed project `project_DP_RTL_implemented` you can load the results by clicking **Implemented Design** in the Navigator.
3. To verify the results were successful, look at the final timing score and view the detailed timing report ([Figure 1-10](#)). Note that the timing score is "0" and that worst case path listed still has positive slack. Also, the image in [Figure 1-10](#) shows the logic of the two `usbEngine` instances highlighted in green and orange. This is just to show the placement was controlled by the `AREA_GROUP` constraints. To highlight the primitives of these two instances, select the two instances in the Netlist view, right-click and select **Highlight Primitives**.

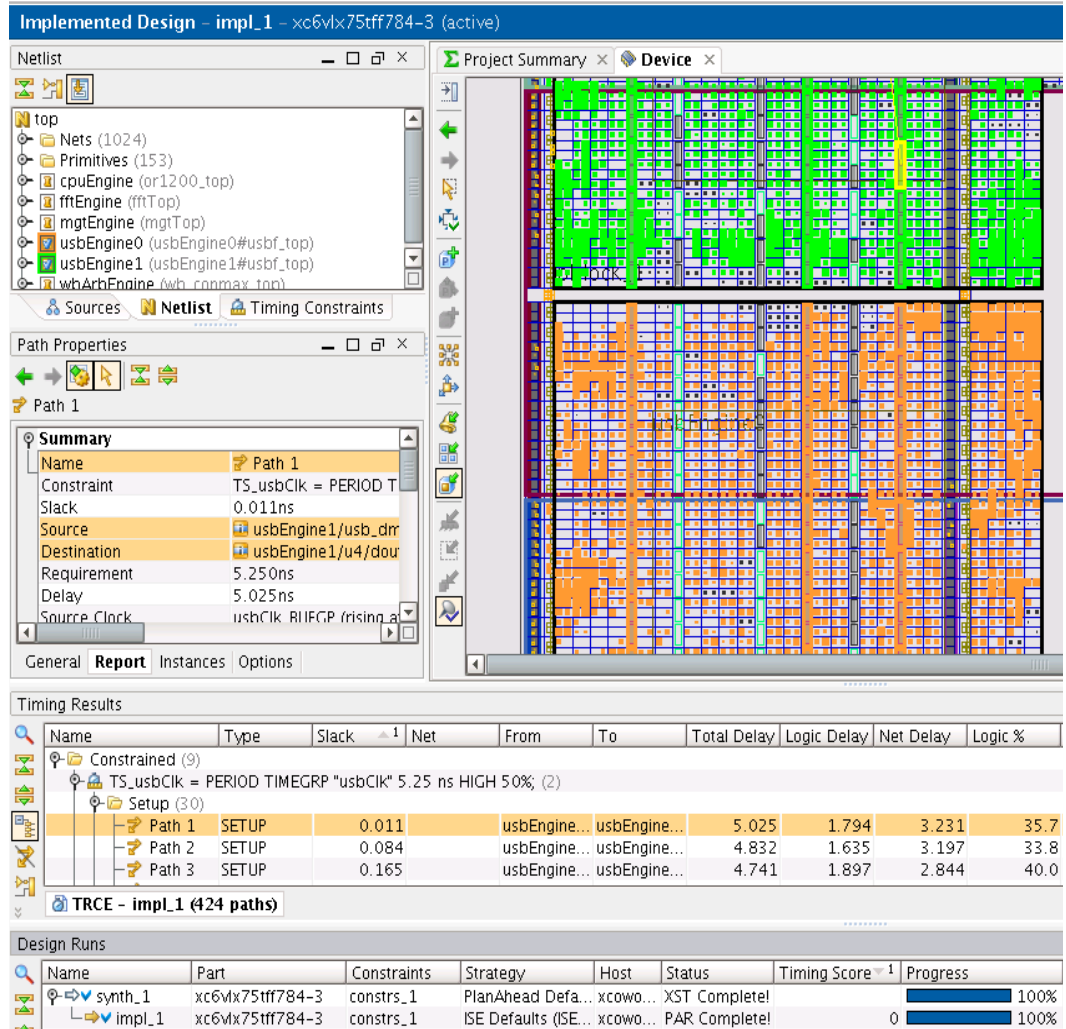


Figure 1-10: Verifying Successful Implementation Results

- In the Flow Navigator, click **Promote Partitions** to promote the synthesis and implementation results (Figure 1-11).

Note that if the RTL Design view is not open when you try to promote, you are asked if it is okay to open the RTL Design view. Click **OK**.

Note: This may also generate the same Critical Warnings as those from [Elaborating an RTL Design](#), page 7.

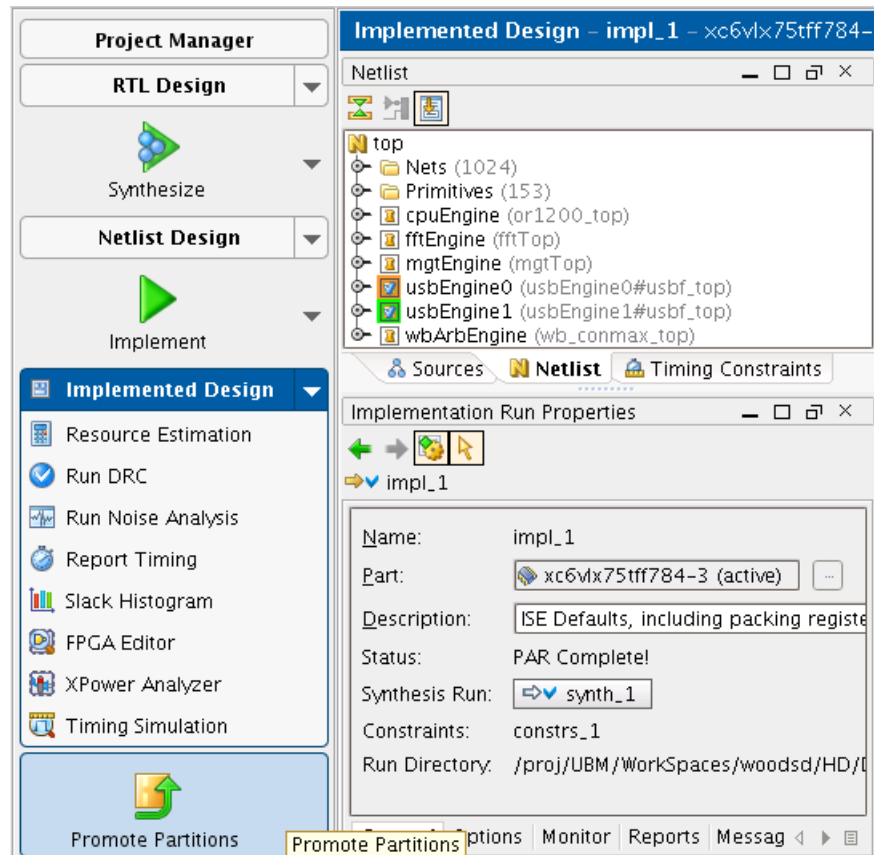


Figure 1-11: Promote Partitions

5. Verify that the two usbEngine module instances are checked for promoting for both synthesis and implementation (Figure 1-12).

Note that the top-level partition is not selected by default. However, this partition could be promoted like any other partition. For this tutorial, you update the top-level partition, so there is no need to promote it here.

6. Click **OK** to promote the two usbEngine partitions. Optionally, you can enter a description about the promoted data. Also, verify that the **Automatically manage Partition action and import location** check box is checked. This allows the PlanAhead software to update the partition state to import location for the next synthesis and implementation run. If this box is not checked, it is the users responsibility to manage these attributes.

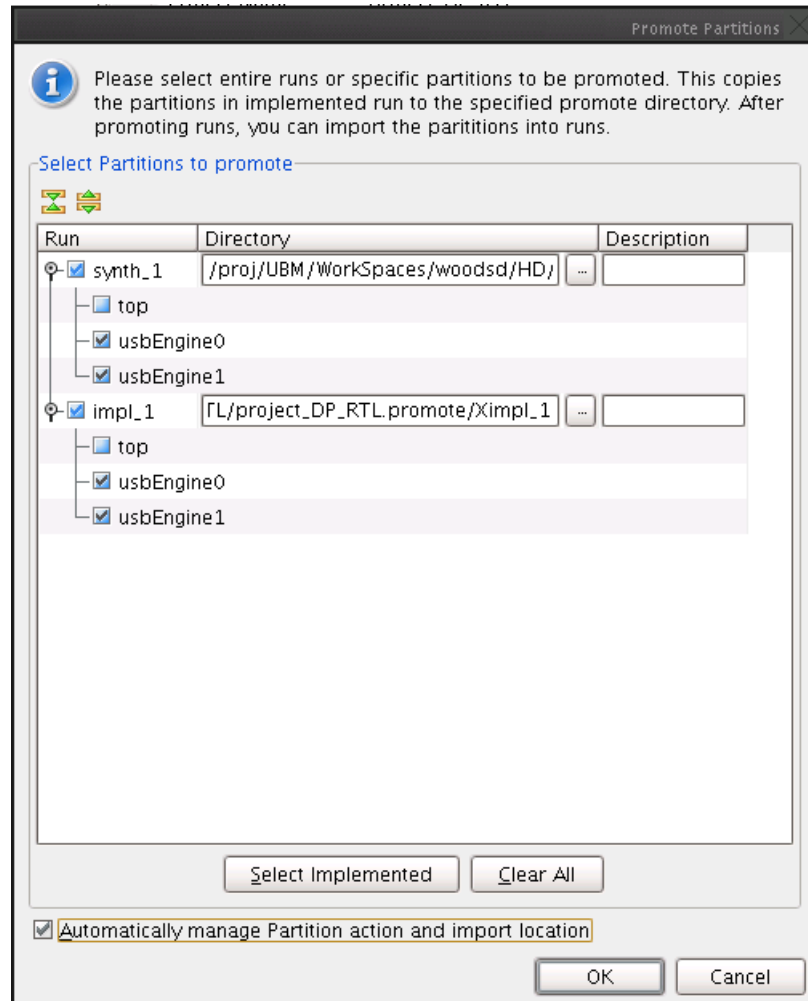


Figure 1-12: Promote Partitions Dialog Box

7. Explore the changes caused by promoting partitions (Figure 1-13):
 - In the **RTL Design** view, you can see a **Promoted Partitions** tab.
 - In the **Synthesis and Implementation Settings** dialog box, the **Specify Partitions** box now shows the Action on the usbEngine instances as Import.

Note: To access the Synthesis or Implementation Settings, click on the pull-down menu on the right side of the Synthesize or Implement buttons, respectively. The Specify Partitions dialog box can be accessed from there.

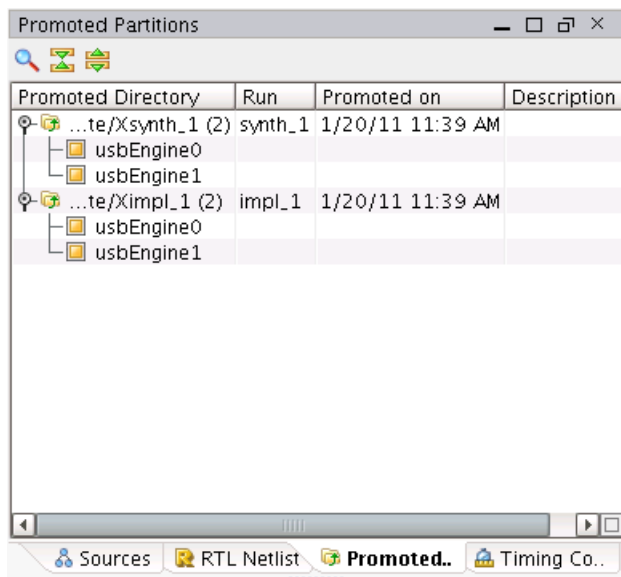


Figure 1-13: The Promoted Partitions

Step 5: Modifying the RTL for Top

Now that the successful synthesis and implementation results have been promoted, the next step is make a change to the design. This could be a new feature, a bug fix, or a pipeline register. For this tutorial we are going to make a very simple change to a module that belongs to the Top partition. We then re-synthesize and re-implement on the changed partitions (in this case Top), while preserving the two timing critical usbEngine partitions.

Changing the VHDL File or1200_defins.v.

1. Click **Project Manager** in the Flow Navigator to open the **Project Manager** view.
2. Scroll through the list of files and locate the Verilog header file `or1200_defins.v` in the **Sources** window.
3. Double-click `or1200_defins.v` to open it in the Text Editor ([Figure 1-14](#)).

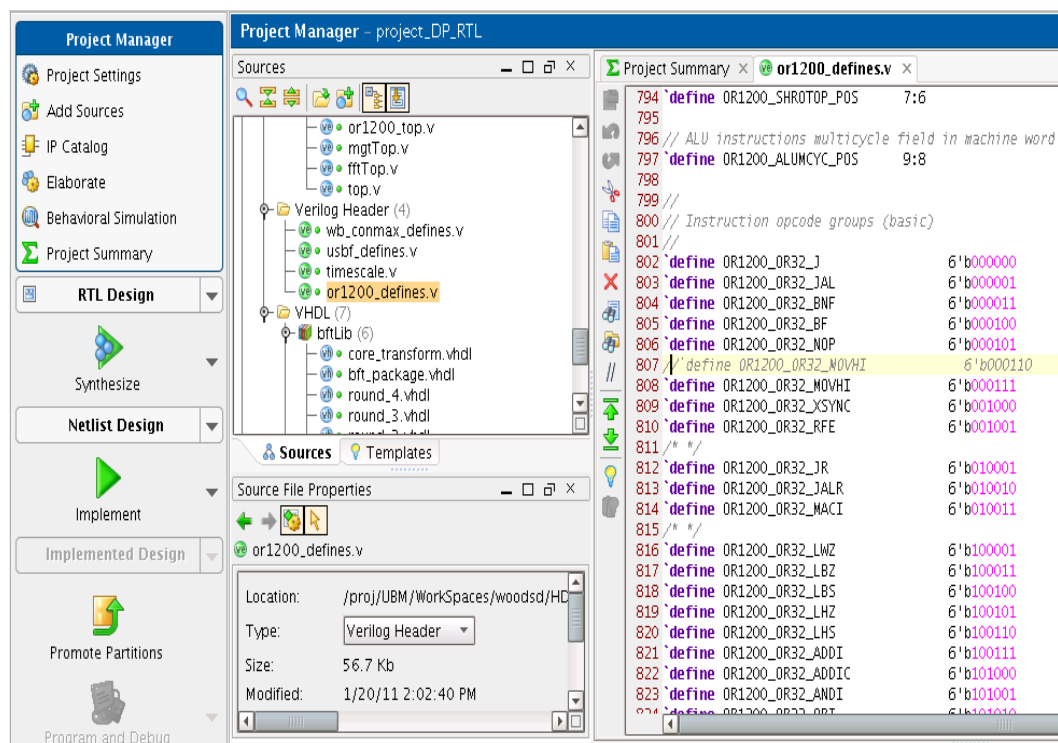


Figure 1-14: Updating `or1200_defins.v`

4. Scroll down to line 807, and comment out this lines by inserted two slashes ("`//`") at the beginning of the line.


```
//`define OR1200_OR32_MOVHI                6'b000110
```
5. Next, uncomment line 808 by removing the two slashes ("`//`") at the beginning of the line.


```
`define OR1200_OR32_MOVHI                6'b000111
```
6. To save the changes, right-click in the Text Editor, and select **Save File**.

You should now see the status "Synthesis & Implementation Out-of-date" in the upper right hand corner. The software recognizes that a source file has been modified, and that the current synthesis and implementation results do not reflect the latest version of the design.

Step 6: Rerunning Synthesis and Implementation While Importing

At this point you have defined partitions, defined Pblocks, synthesized and implemented the design, promoted the two usbEngine instances, and made a change to the top-level partition. You can now reimplement the modified top-level partition while maintaining an exact copy of the placement and routing results on the two USB cores.

To remove the implementation runtime from this tutorial, a completed project with synthesis and implementation results can be found at: `<Extract_Dir>/Projects/project_DP_RTL_imported/project_DP_RTL_imported.ppr`

Select **File > Open Project** to skip this step and open the completed results.

Verifying the Synthesis Partition Attributes

1. From the Navigator, click the pull-down menu on the **Synthesize** button and choose **Synthesis Settings**.
2. From the **Synthesis Settings** dialog box, click the **Specify Partitions** button to open this window (Figure 1-15).
3. Verify that the top-level partition is set to **Implement** and that the two usbEngine partitions are set to **Import**. Click **OK**.

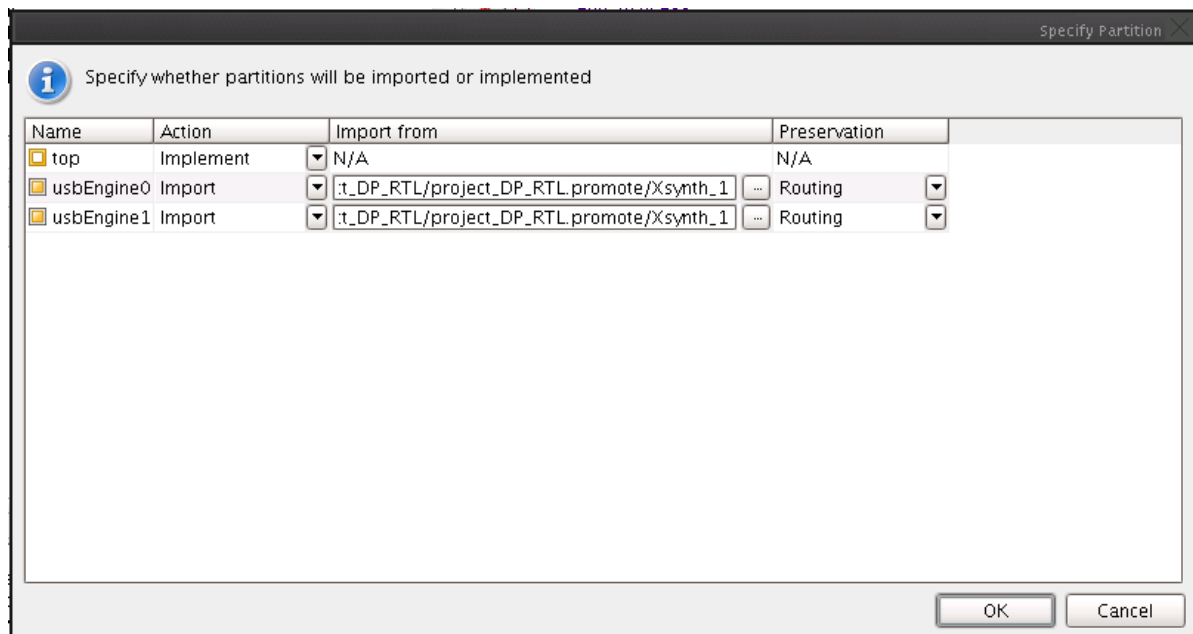


Figure 1-15: Verifying Partitions Attributes

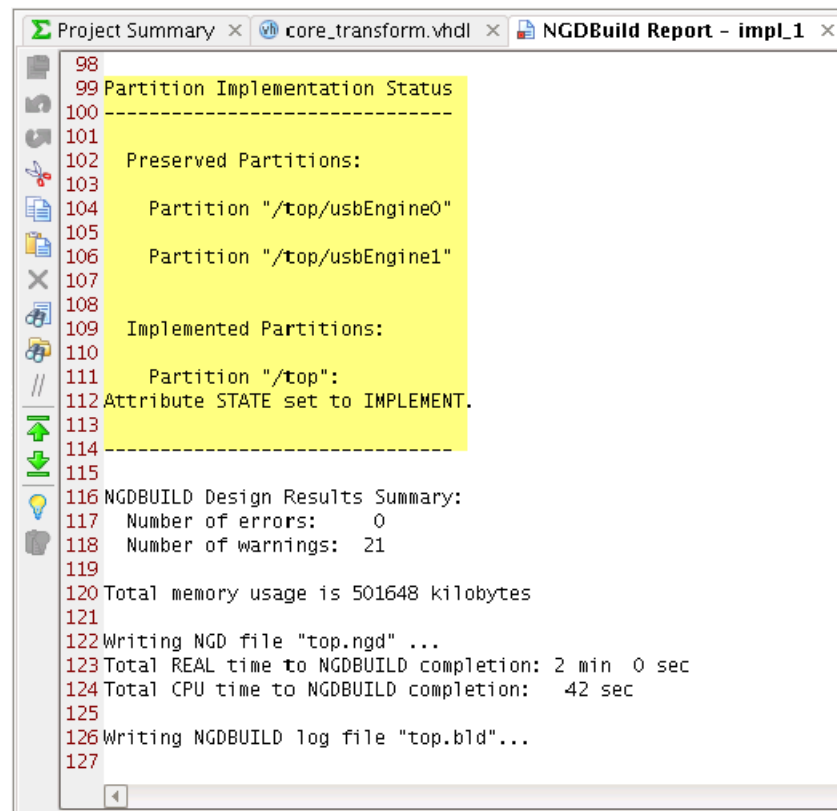
4. Click on **Save** to exit the Synthesis Settings dialog box.

Verifying the Implementation Partition Attributes

1. Repeat Step 1-4 above for the Implementation Settings. Use the pull-down menu on the **Implement** button to access the **Implementation Settings**.

Running Synthesis and Implementation

1. In the Navigator, click **Implement** to launch the implementation run. Because Synthesis is out-of-date, you are prompted to launch synthesis first. Click **Yes** to launch **Synthesis** and **Implementation**.
2. Verify that the two usbEngine partitions have been imported and that all timing constraints have been met. Look at the **Partition Status** section in the NGDBuild, Map, or PAR reports (Figure 1-16).



```
98
99 Partition Implementation Status
100 -----
101
102 Preserved Partitions:
103
104 Partition "/top/usbEngine0"
105
106 Partition "/top/usbEngine1"
107
108 Implemented Partitions:
109
110 Partition "/top":
111 Attribute STATE set to IMPLEMENT.
112 -----
113
114
115
116 NGDBUILD Design Results Summary:
117 Number of errors: 0
118 Number of warnings: 21
119
120 Total memory usage is 501648 kilobytes
121
122 Writing NGD file "top.ngd" ...
123 Total REAL time to NGDBUILD completion: 2 min 0 sec
124 Total CPU time to NGDBUILD completion: 42 sec
125
126 Writing NGDBUILD log file "top.bld"...
127
```

Figure 1-16: Partition Implementation Status in Report Files

Conclusion

In this tutorial, you defined partitions, created Pblock constraints, and ran synthesis and implementation on the design. You then verified that timing was met and promoted the successful results to allow for importing in future iterations. The top module was updated requiring synthesis and implementation to be re-run. However, because the USB cores were not modified, they were imported, and they maintained identical placement and routing results. The output guaranteed timing results on two large, timing-critical cores for all future iterations of the design (assuming no changes to the usbEngine instances).

Additional Resources

Xilinx Resources

- *ISE Design Suite: Installation and Licensing Guide (UG798):*
http://www.xilinx.com/support/documentation/sw_manuals/xilinx13_1/iil.pdf
- *ISE Design Suite 13: Release Notes Guide (UG631):*
http://www.xilinx.com/support/documentation/sw_manuals/xilinx13_1/irn.pdf
- **Xilinx® Documentation:**
<http://www.xilinx.com/support/documentation.htm>
- **Xilinx Global Glossary:**
http://www.xilinx.com/support/documentation/sw_manuals/glossary.pdf
- **Xilinx Support:** <http://www.xilinx.com/support.htm>
- **Video Demonstrations:**
http://www.xilinx.com/products/design_resources/design_tool/resources/index.htm

PlanAhead Documentation

- *PlanAhead User Guide (UG632):*
http://www.xilinx.com/support/documentation/sw_manuals/xilinx13_1/PlanAhead_UserGuide.pdf
- **PlanAhead Methodology Guides:**
http://www.xilinx.com/support/documentation/dt_planahead_planahead13-1_userguides.htm
 - *Hierarchical Design Methodology Guide (UG748)*
http://www.xilinx.com/support/documentation/sw_manuals/xilinx13_1/Hierarchical_Design_Methodology_Guide.pdf
- **PlanAhead Tutorials:**
http://www.xilinx.com/support/documentation/dt_planahead_planahead13-1_tutorials.htm
 - *Design Analysis and Floorplanning for Performance (UG676)*
http://www.xilinx.com/support/documentation/sw_manuals/xilinx13_1/PlanAhead_Tutorial_Design_Analysis_Floorplan.pdf

