

# Design Preservation Tutorial

## *PlanAhead Design Tool*

UG747 (v 13.4) January 03, 2012





Xilinx is disclosing this user guide, manual, release note, and/or specification (the “Documentation”) to you solely for use in the development of designs to operate with Xilinx hardware devices. You might not reproduce, distribute, republish, download, display, post, or transmit the Documentation in any form or by any means including, but not limited to, electronic, mechanical, photocopying, recording, or otherwise, without the prior written consent of Xilinx. Xilinx expressly disclaims any liability arising out of your use of the Documentation. Xilinx reserves the right, at its sole discretion, to change the Documentation without notice at any time. Xilinx assumes no obligation to correct any errors contained in the Documentation, or to advise you of any corrections or updates. Xilinx expressly disclaims any liability in connection with technical support or assistance that might be provided to you in connection with the Information.

THE DOCUMENTATION IS DISCLOSED TO YOU “AS-IS” WITH NO WARRANTY OF ANY KIND. XILINX MAKES NO OTHER WARRANTIES, WHETHER EXPRESS, IMPLIED, OR STATUTORY, REGARDING THE DOCUMENTATION, INCLUDING ANY WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NONINFRINGEMENT OF THIRD-PARTY RIGHTS. IN NO EVENT WILL XILINX BE LIABLE FOR ANY CONSEQUENTIAL, INDIRECT, EXEMPLARY, SPECIAL, OR INCIDENTAL DAMAGES, INCLUDING ANY LOSS OF DATA OR LOST PROFITS, ARISING FROM YOUR USE OF THE DOCUMENTATION.

© Copyright 2011 Xilinx Inc. All Rights Reserved. XILINX, the Xilinx logo, the Brand Window and other designated brands included herein are trademarks of Xilinx, Inc. All other trademarks are the property of their respective owners. The PowerPC name and logo are registered trademarks of IBM Corp., and used under license. All other trademarks are the property of their respective owners.

# *Table of Contents*

---

Software Requirements .....	4
Hardware Requirements.....	4
Tutorial Design Description .....	4
Step 1: Opening an Existing PlanAhead RTL Project and Elaborating the RTL Design ...	6
Step 2: Setting Partitions and Drawing Pblocks .....	8
Step 3: Synthesizing and Implementing the Design .....	12
Step 4: Promoting Successfully Implemented Partitions .....	16
Step 5: Modifying the RTL for Top.....	20
Step 6: Rerunning Synthesis and Implementation While Importing .....	21
Conclusion .....	23

# Design Preservation Tutorial

---

This tutorial provides an overview of the design preservation flow. In this tutorial, you will:

- Define partitions and Pblocks on an elaborated Register Transfer Level (RTL) design.
- Synthesize using the Xilinx® Synthesis Technology (XST) incremental synthesis software.
- 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.

The objective of this tutorial is to familiarize you with the partitions and the design preservation flow using the PlanAhead design tool. Many of the PlanAhead tool analysis features are covered in more detail in other tutorials, and not every command or command option is covered.

## Software Requirements

The PlanAhead tool installs with the ISE® Design Suite software. Before starting the tutorial, ensure that the PlanAhead tool 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)* at [http://www.xilinx.com/support/documentation/sw\\_manuals/xilinx13\\_4/iil.pdf](http://www.xilinx.com/support/documentation/sw_manuals/xilinx13_4/iil.pdf).

## Hardware Requirements

Xilinx recommends a minimum of 2 GB of RAM when using the PlanAhead tool 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 set of synthesis results was created using a standard top-down synthesis flow to be used in a flat implementation.
- The second set of synthesis results was created using an incremental synthesis flow to be used in the Design Preservation flow. This set has separate netlists for the 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 xc6vlx75t 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.

Download the PlanAhead\_Tutorial.zip file from the Xilinx website:

[http://www.xilinx.com/support/documentation/dt\\_planahead\\_planahead13-4\\_tutorials.htm](http://www.xilinx.com/support/documentation/dt_planahead_planahead13-4_tutorials.htm)

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.

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

This tutorial uses pre-existing PlanAhead tool projects to simplify the steps and focus on the design preservation aspects of the tutorial. For a real design, you would use the New Project Wizard to create either an RTL or netlist based project. The PlanAhead tool (v. 13.1 or newer) supports RTL projects with partitions.

### Opening an Existing PlanAhead tool RTL Project

To open an existing PlanAhead tool RTL project:

1. Open the PlanAhead tool.
  - a) On Windows, double-click the Xilinx® PlanAhead 13 Desktop icon, or select:  
**Start > Programs > Xilinx ISE Design Suite 13.4 > PlanAhead > PlanAhead**
  - b) On Linux, go to the following directory and type **PlanAhead**:  
**<ISE\_install\_dir>/PlanAhead\_Tutorial/Tutorial\_Created\_Data**
2. From the Getting Started page, click **Open Project**.
3. In the <Extract\_Dir> directory, open the project file located at:  
`./Projects/project_DP_RTL/project_DP_RTL.ppr`

### Viewing Source Files

When the project opens, the Project Manager view is visible. You can view the following design files in the Sources window:

- VHDL source files
- Verilog source files
- A User Constraints File (UCF) named `top_full.ucf`. This file contains timing constraints and I/O pin locations.

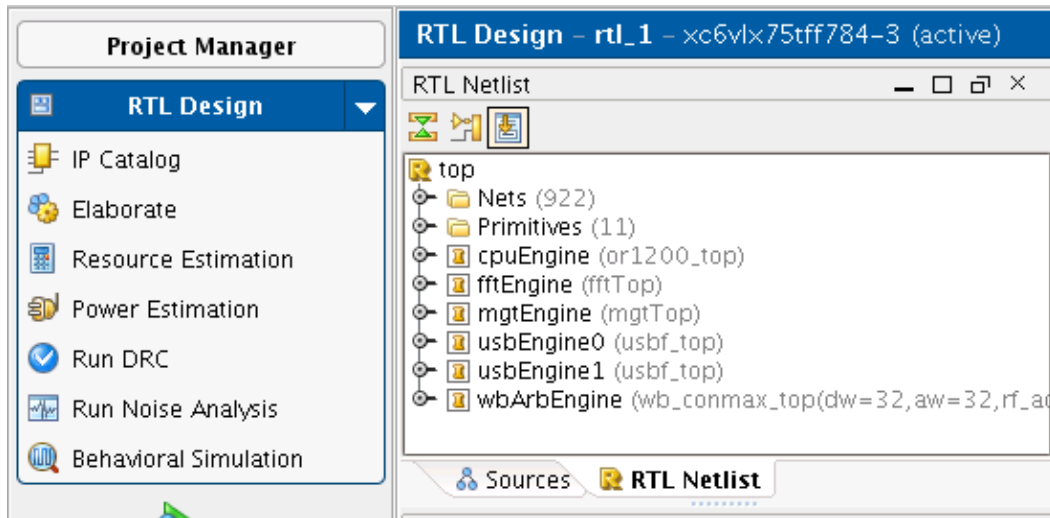
### Elaborating an RTL Design

You must use the RTL Design view to define partitions in an RTL project. When the RTL Design view is opened:

- The RTL code is elaborated.
  - The design hierarchy is displayed.
- Use this pre-synthesized view of the design to define partitions and create constraints.

To elaborate an RTL design:

1. Select **Flow > RTL Design**, or click **RTL Design** in the Flow Navigator.



**Figure 1: Opening the RTL Design View**

2. Ignore the Critical Warnings message.

When the RTL Design view opens, it parses the UCF. Because the RTL design view is a pre-synthesized version of the design, there are instances with constraints in the UCF that do not exist in the elaborated design (such as I/O buffers). Because of this, the message is expected. Ignore it.

3. Click **OK**.

## Step 2: Setting Partitions and Drawing Pblocks

Because the **usbEngine** instances have already been identified as timing-critical modules, it would be advantageous to preserve the successful implementation results of these instances. However, this fact alone does not make these instances good candidates for partitions.

The **usbEngine** instances are good choices for partitions because:

- They are logically isolated from the rest of the design.
- They have reasonable interface timing (registered inputs and outputs).

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, see *the Hierarchical Design Methodology Guide (UG748)*.

You can floorplan partitioned instances like any other instances. 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 the **usbEngine** along the left side of the device. The steps below show you how to create appropriate Pblock constraints for the two **usbEngine** instances.

### Setting Partitions for the two usbEngine Instances

To set partitions for the two **usbEngine** instances:

1. From the RTL Design view, select the two **usbEngine** instances from the Netlist tab.
2. Right-click.
3. Select **Set Partition**.

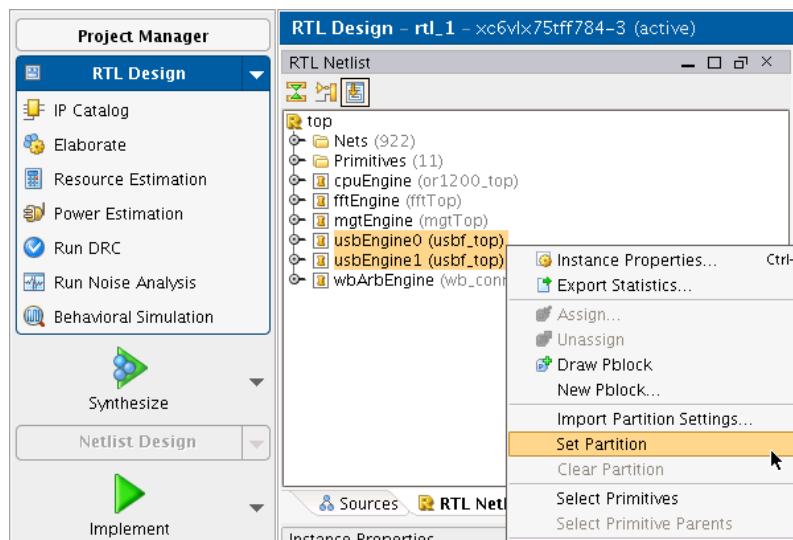


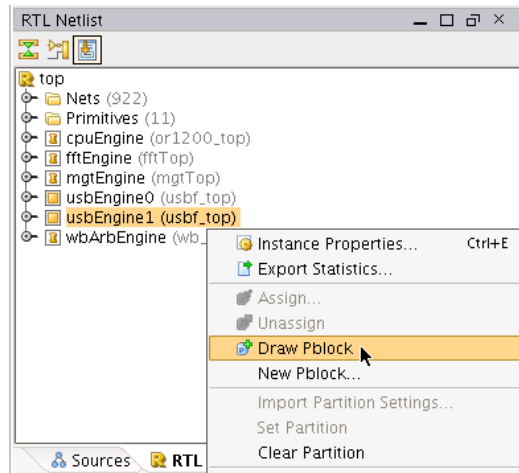
Figure 2: Setting Partitions on usbEngine Instances

## Drawing Pblocks for the Two usbEngine Instances

This step does not affect synthesis results. You can also perform this step post-synthesis in the Netlist Design view. There may be some advantages to doing so, such as resource estimation to help size the Pblocks.

To draw Pblocks for the two **usbEngine** instances:

1. From the Netlist window, select and right-click **usbEngine1**.
2. Select **Draw Pblock**.



**Figure 3: Selecting the Draw Pblock Tool**

3. With the Draw Pblock tool active, move the cursor to the Device window.
4. Left click the top-left corner of the device where the CLB components start.
5. Without releasing the left mouse button, drag to create a rectangle covering most of the top-left quadrant of the device. See the figure below (*Pblock Rectangle for usbEngine1*).
6. In the New Pblock dialog box, verify that the SLICE and RAMB36 grids are selected.
7. Deselect other resources that are not needed. See the figure below (*Pblock Rectangle for usbEngine1*).
8. Verify that the number of available RAMB36 components (shown in parentheses) is 36.

If the rectangle does not fully cover the region shown in the figure below (*Pblock Rectangle for usbEngine1*), this number might be less, and the design will fail to place.

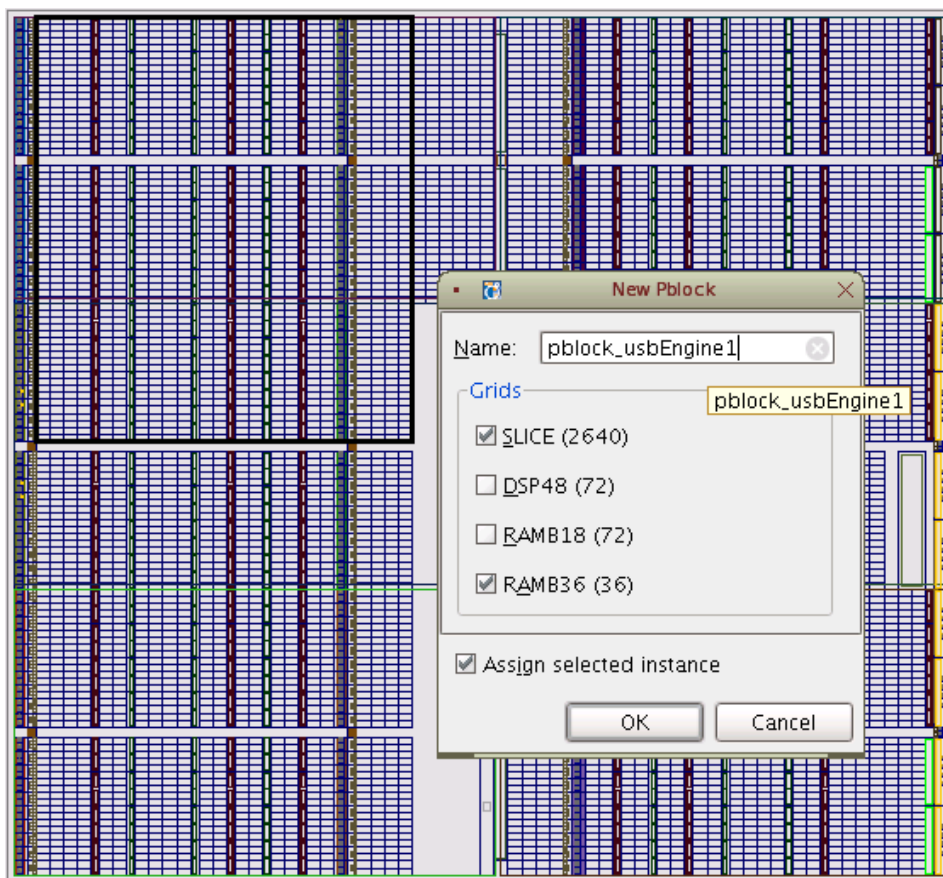


Figure 4: Pblock Rectangle for usbEngine1

9. Click **OK**.
10. Confirm the number of available RAMB36 resources.

If the Pblock rectangle does not completely cover the RAMB36 resources as shown in the figure below (*Completed Floorplan for Pblocks usbEngine0 and usbEngine1*), 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.

11. Repeat the steps above for **usbEngine0** on the bottom-left quadrant.

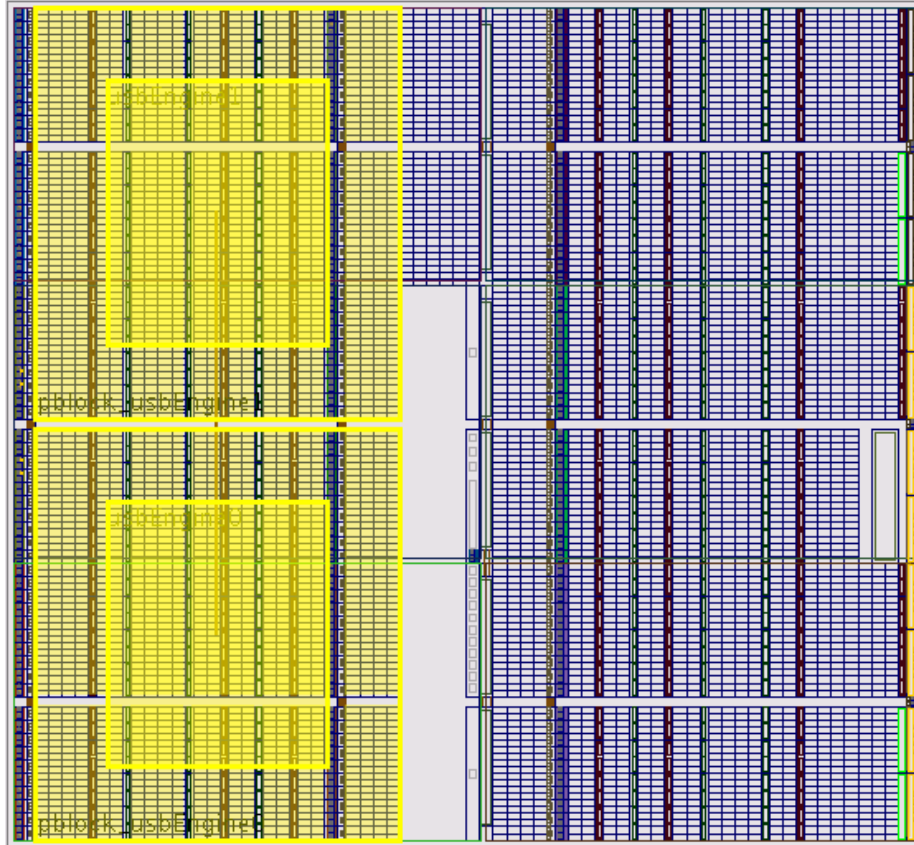


Figure 5: Completed Floorplan for Pblocks usbEngine0 and usbEngine1

## Step 3: Synthesizing and Implementing the Design

Now that all partitions have been defined, and all necessary constraints have been created, you can run synthesis and implementation. Because partitions were defined in the elaborated Hardware Description Language (HDL) design, XST does the following:

- Detects the partitions.
- Runs incremental flow.
- Creates an individual NGC file for each defined partition.

### NGC Files

The individual NGC files for each defined partition are uniquely named. Unique naming allows multiple instances of a module to be synthesized with different parameters. In this case, synthesis generates the following NGC files in the `<project_name>.runs/synth_1` directory:

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

### Running Synthesis outside the PlanAhead tool

If you prefer a bottom-up synthesis, or a third party incremental synthesis flow, run synthesis outside the PlanAhead tool.

Use a PlanAhead netlist project instead of the RTL project shown in this tutorial.

### Running Synthesis in the PlanAhead tool

To run synthesis in the PlanAhead tool:

1. Click **Synthesize** in the Navigator pane.

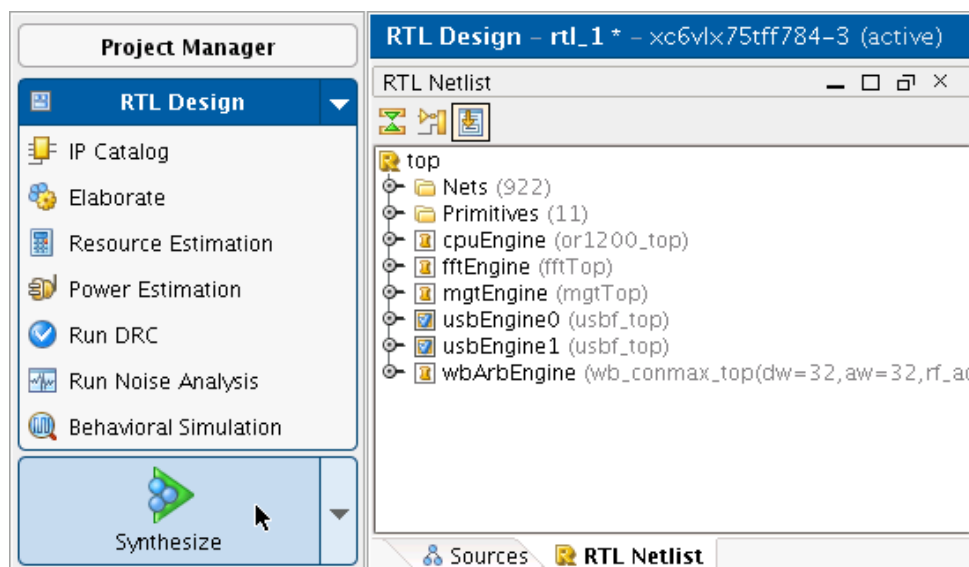


Figure 6: Launch XST Incremental Synthesis

2. To set specific XST options, or to check the settings on the partitions, click the **Synthesize** button pull-down menu.
3. Select **Synthesis Settings**.  
Since the defaults are acceptable, click **Synthesize**.
4. Click **Save**.
5. When synthesis completes a dialog will appear with several options. Choose **Open Netlist Design** and click **OK**.

## Running a DRC on Partitions

While you can run a Design Rule Check (DRC) on the RTL Design, there are limitations on the parameters that the software can check at this stage. Xilinx recommends that you run a DRC before launching implementation. To run a DRC, load the Netlist Design view, then run partition-specific DRCs.

To run a DRC on partitions:

1. Click **Netlist Design** to open the Netlist Design view if not already open.  
Opening the Netlist Design view loads the synthesis results and allows for additional DRC checking.
2. In the Navigator under Netlist Design, click **Run DRC** (or select **Tools > Run DRC**).
3. From the Run DRC dialog box, deselect all rules except **Partition**.

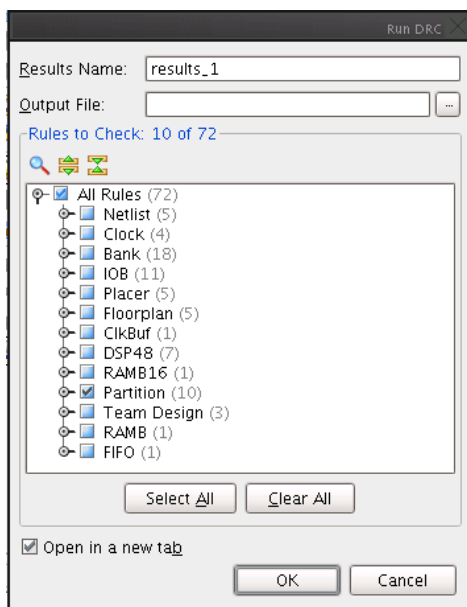


Figure 7: Run Partition Design Rule Check

4. Click **OK**.

## DRC Advisory Messages

The DRC returns minor Advisory messages. The PlanAhead tool can report the following message types for the DRC rules:

- Advisory
- Warning
- Error
- Fatal

Ignore the Advisory messages for this tutorial. In an actual design, investigate all DRC messages, and correct any serious issues.

## Running Implementation in the PlanAhead tool

You are now ready to run implementation in the PlanAhead tool. To implement this design with partitions, the only additional steps required were to:

- Define the partitions.
- Run a DRC check.

Since this design was already created with hierarchy in mind, you did not have to alter the design to make it 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, see the *Hierarchical Design Methodology Guide (UG748)*.

## Skipping Implementation

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
```

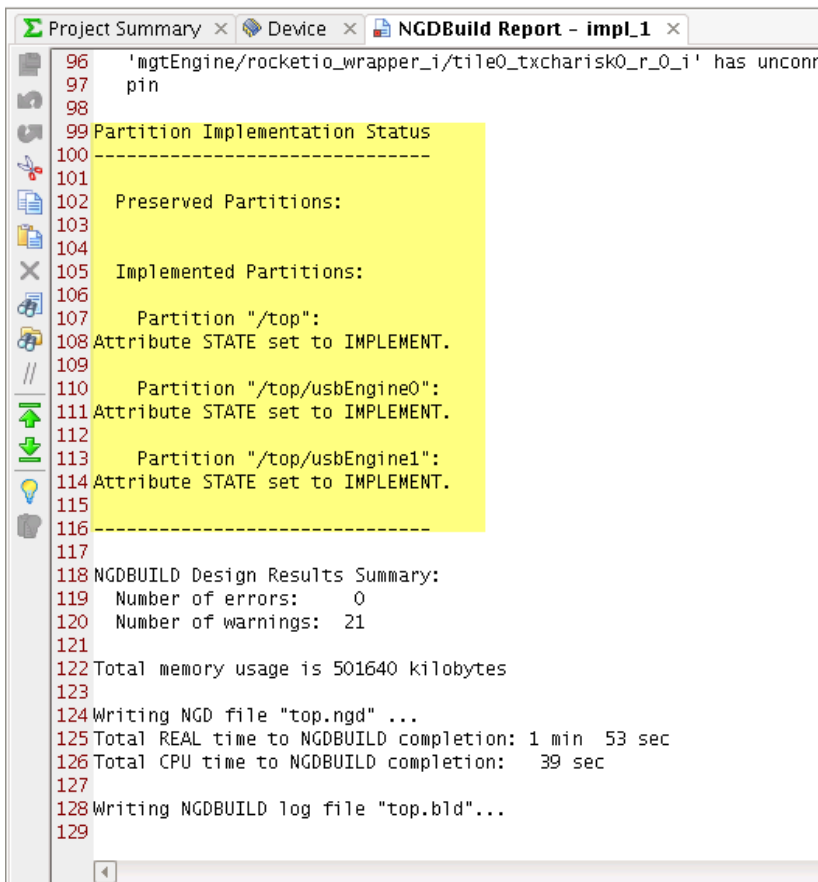
Select **File > Open Project** to skip this step and open the completed results.

If you skip this step, go to *Step 4: Promoting Successfully Implemented Partitions*.

## Running Implementation

To run implementation in the PlanAhead tool:

1. In the Flow Navigator, click **Implement**.
2. Click the **Report** tab to see a list of implementation reports. Each report is available as the process finishes.
3. Once NGDBuild finishes, double-click **NGDBuild Report**.
4. Scroll to the bottom of the report to view the partition information.

The image shows a screenshot of the 'NGDBuild Report - impl\_1' window in a software application. The window has a title bar with three tabs: 'Project Summary', 'Device', and 'NGDBuild Report - impl\_1'. The main content area displays a text-based report with line numbers on the left. A yellow highlight covers the 'Partition Implementation Status' section, which lists 'Preserved Partitions' and 'Implemented Partitions'. The 'Implemented Partitions' section lists three partitions: '/top', '/top/usbEngine0', and '/top/usbEngine1', each with the attribute 'STATE set to IMPLEMENTED.'. Below this, a 'NGDBUILD Design Results Summary' section shows 'Number of errors: 0' and 'Number of warnings: 21'. The report also includes 'Total memory usage is 501640 kilobytes', 'Writing NGD file "top.ngd" ...', 'Total REAL time to NGDBUILD completion: 1 min 53 sec', 'Total CPU time to NGDBUILD completion: 39 sec', and 'Writing NGDBUILD log file "top.bld"...'. The window has a standard toolbar on the left and a scroll bar at the bottom.

**Figure 8: Implementation Status in Report Files**

This partition information:

- Appears in every report (NGDBuild, Map, and PAR).
- Allows you to easily verify the status of all partitions on a given run.

## Step 4: Promoting Successfully Implemented Partitions

Once an implementation is successful, you can promote the results. Promoting the results makes a copy of the implementation directory in:

```
<project_name>.promote\X<run_name>
```

For example:

```
project_DP_RTL.promote\Ximpl_1
```

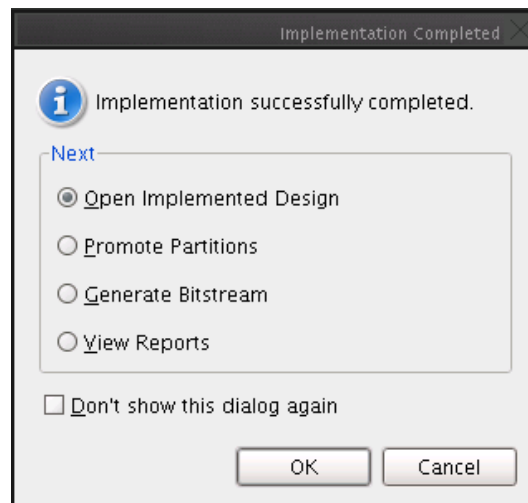
The PlanAhead tool:

- Keeps track of the latest promoted run.
- Changes the state and import location of any promoted partitions.

You can also manage these processes manually.

### Promoting the Successful Implementation Results

Once implementation has completed, the Implementation Completed dialog box opens.



**Figure 9: Implementation Completed Dialog Box**

To promote the successful implementation results:

1. Select **Open Implemented Design** and click **OK**.

Other options are also available. For example, you could promote the results by selecting **Promote Partitions** option. However, this tutorial uses another method to promote the results.

If the Implementation Completed dialog box did not open, or if you opened the completed project **project\_DP\_RTL\_implemented**, load the results by clicking **Implemented Design** in the Navigator.

2. To verify that the results were successful, review the final timing score and the detailed timing report. See the figure below (*Verifying Successful Implementation Results*).
  - The timing score is **0**.
  - The worst case path listed still has positive slack.

The following figure (*Verifying Successful Implementation Results*) shows the logic of the two **usbEngine** instances. This shows that the placement was controlled by the AREA\_GROUP constraints.

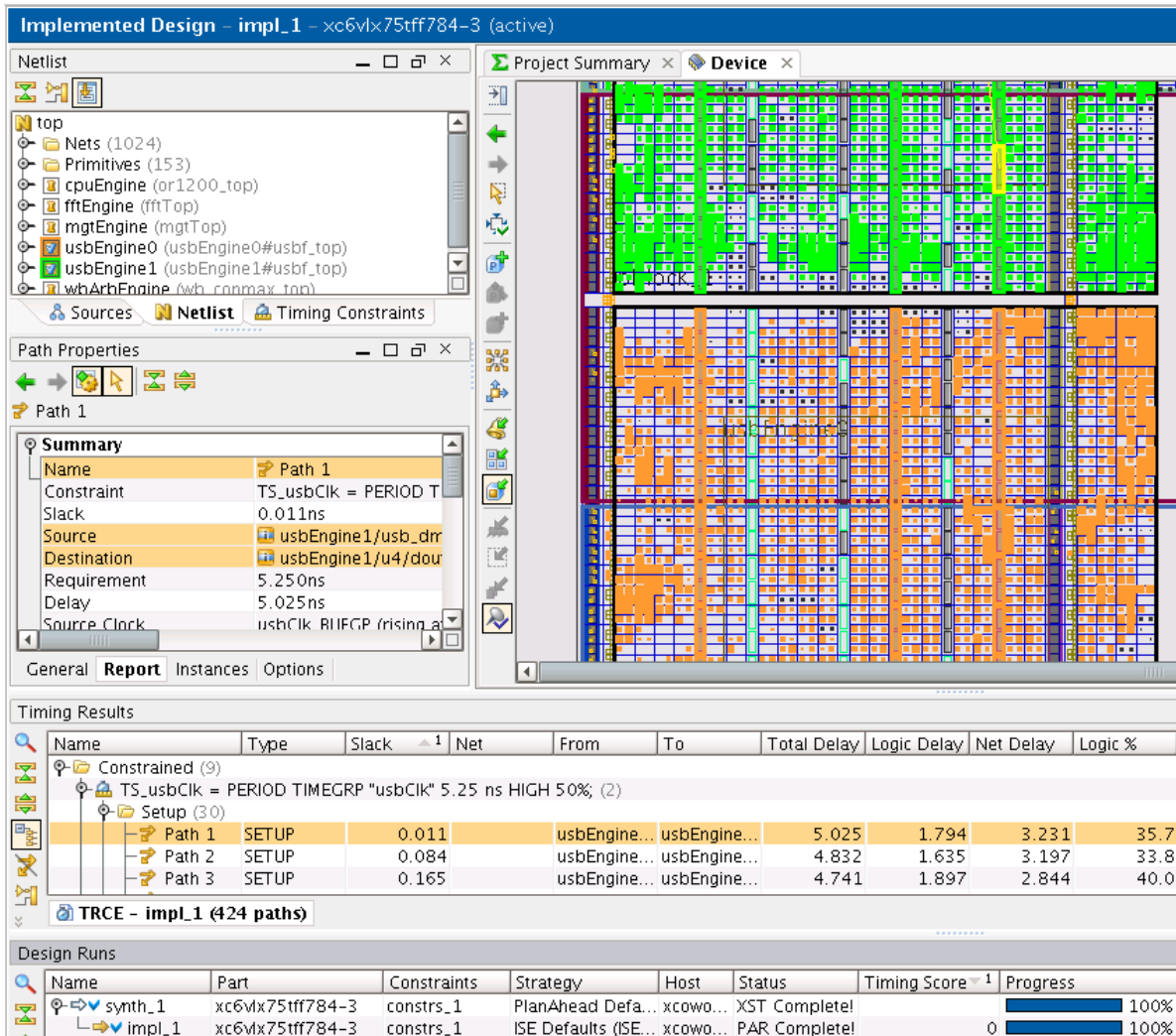


Figure 10: Verifying Successful Implementation Results

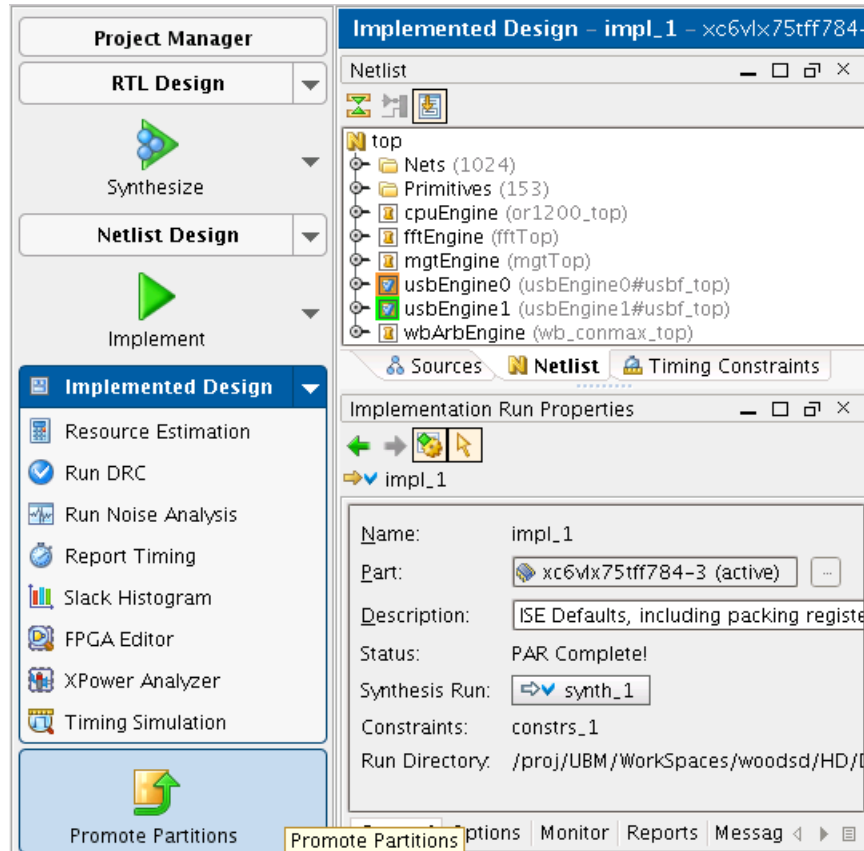
3. To highlight the primitives of the two **usbEngine** instances:
  - a) Select the two instances in the Netlist view.
  - b) Right-click.
  - c) Select **Highlight Primitives**.

- In the Flow Navigator, click **Promote Partitions** to promote the synthesis and implementation results. See the figure below (*Promote Partitions*).

If the RTL Design view is not open when you try to promote, you are asked if you want to open the RTL Design view.

- Click **OK**.

This may also generate the same Critical Warnings as those from *Elaborating an RTL Design*.



**Figure 11: Promote Partitions**

- Verify that the two **usbEngine** module instances are checked for promoting for both synthesis and implementation. See the figure below (*Promote Partitions Dialog Box*).

The top-level partition is not selected by default. However, you can promote this partition like any other partition. For this tutorial, since you update the Top partition, there is no need to promote it now.

- Click **OK** to promote the two **usbEngine** partitions.
- Enter a description about the promoted data (optional).
- Verify that **Automatically Manage Partition Action and Import Location** is checked.

Checking **Automatically Manage Partition Action and Import Location** allows the PlanAhead tool to update the partition state to import location for the next synthesis and implementation run. If this option is not checked, you must manage these attributes yourself.

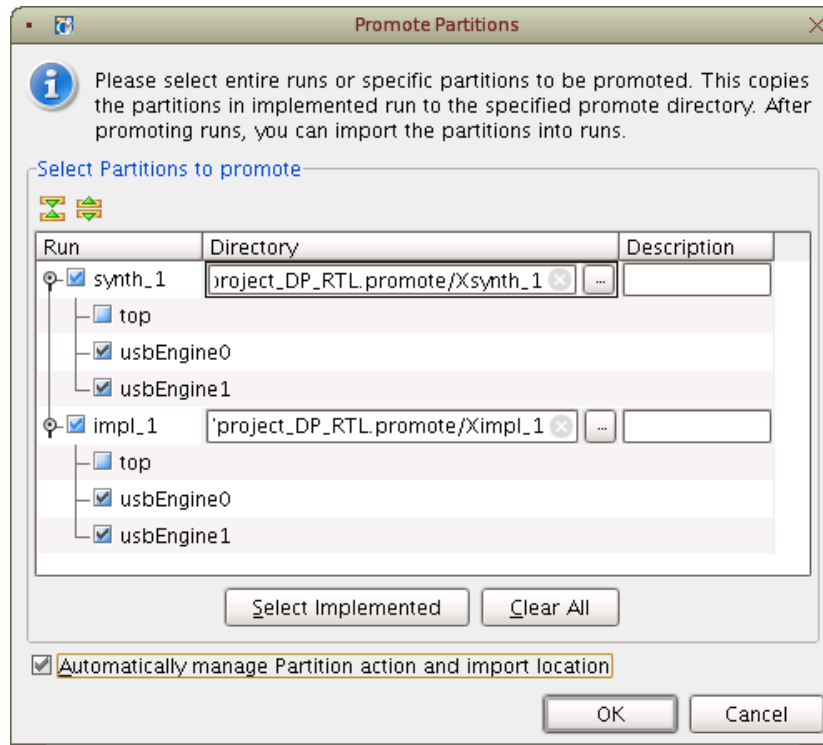


Figure 12: Promote Partitions Dialog Box

10. Review the changes caused by promoting partitions. See the figure below (*Promoted Partitions*).
  - 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**.
11. To access the synthesis settings, click the **Synthesize** button pull-down menu. To access the implementation settings, click the **Implement** button pull-down menu. The Specify Partitions dialog box can be accessed from there.

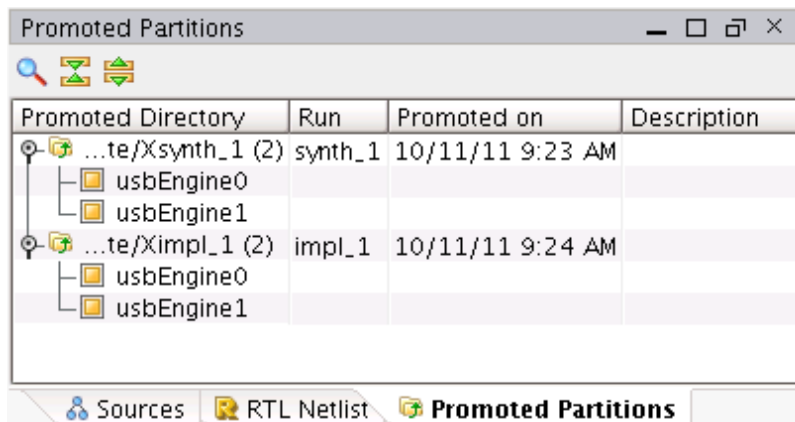


Figure 13: Promoted Partitions

## Step 5: Modifying the RTL for Top

Now that you have promoted the successful synthesis and implementation results, you now make a change to the design. Such a change could be, for example, a new feature, a bug fix, or a pipeline register.

This tutorial makes a simple change to a module belonging to the Top partition. You then re-synthesize and re-implement on the changed partitions (in this case Top), while preserving the two timing-critical **usbEngine** partitions.

### Changing the Verilog File

To change the Verilog 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 to 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.

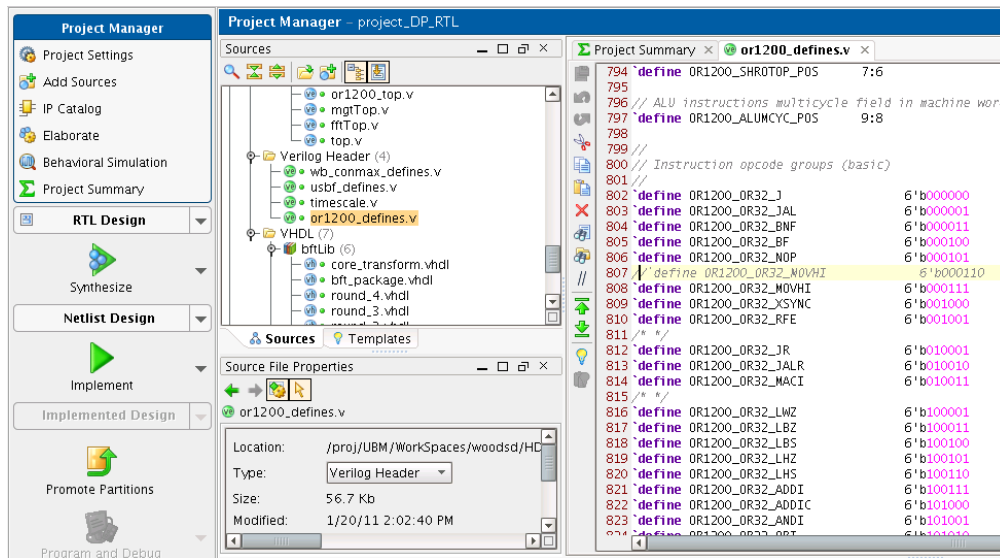


Figure 14: Updating `or1200_defins.v`

4. Insert two slashes (//) at the beginning of the line to comment out line 807.
 

```
//`define OR1200_OR32_MOVHI 6'b000110
```
5. Remove the two slashes (//) at the beginning of the line to uncomment line 808.
 

```
`define OR1200_OR32_MOVHI 6'b000111
```
6. Right-click. And select **Save File**.

**Synthesis & Implementation Out-of-Date** appears in the upper right hand corner. The software recognizes that:

- A source file has been modified.
- The current synthesis and implementation results do not reflect the latest version of the design.

## Step 6: Rerunning Synthesis and Implementation While Importing

You have now:

- Defined partitions.
- Defined Pblocks.
- Synthesized and implemented the design.
- Promoted the two **usbEngine** instances.
- 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.

### Skipping Implementation

To remove the implementation runtime from this tutorial, a completed project with synthesis and implementation results can be found at:

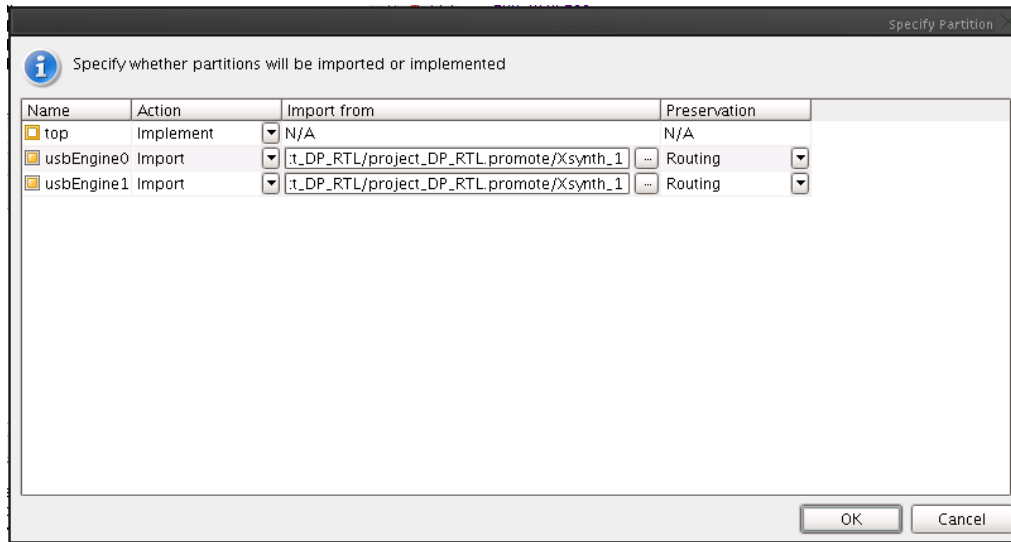
```
<Extract_Dir>/Projects/project_DP_RTL_implemented/project_DP_RTL_implemented.ppr
```

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

### Verifying the Synthesis Partition Attributes

To verify the synthesis partition attributes:

1. From the Navigator, click the **Synthesize** button pull-down menu.
2. Select **Synthesis Settings**.
3. From the Synthesis Settings dialog box, click **Specify Partitions**.
4. Verify that:
  - The top-level partition is set to **Implement**.
  - The two **usbEngine** partitions are set to **Import**.
5. Click **OK**.



**Figure 15: Verifying Partitions Attributes**

6. Click **Save**.

## Verifying the Implementation Partition Attributes

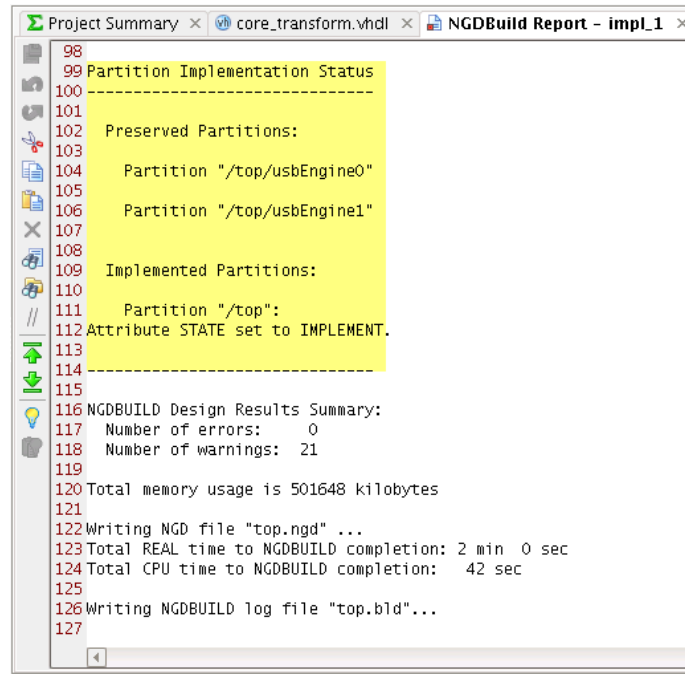
To verify the implementation partition attributes:

1. Select the **Implement** button pull-down menu to access the Implementation Settings.
2. Repeat the steps above for the implementation settings.

## Running Synthesis and Implementation

To run synthesis and implementation:

1. In the Navigator, click **Implement** to launch the implementation run.
2. Because Synthesis is out of date, you are prompted to launch synthesis first. Click **Yes** to launch synthesis and implementation.
3. Review the Partition Status section in the NGDBuild, Map, or PAR reports to verify that:
  - The two **usbEngine** partitions have been imported.
  - All timing constraints have been met.



```
98
99 Partition Implementation Status
100 -----
101
102 Preserved Partitions:
103
104 Partition "/top/usbEngine0"
105
106 Partition "/top/usbEngine1"
107
108
109 Implemented Partitions:
110
111 Partition "/top":
112 Attribute STATE set to IMPLEMENT.
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 16: Partition Implementation Status in Report Files

## Conclusion

In this tutorial, you did the following:

- Defined partitions.
- Created Pblock constraints.
- Ran synthesis and implementation.
- Verified that timing was met.
- Promoted the successful results to allow for importing in future iterations.

The top module was updated. This required synthesis and implementation to be re-run. Because the USB cores were not modified, they were imported and 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).