

Partial Reconfiguration Tutorial

PlanAhead Design Tool

This tutorial document was last validated using the following software version: ISE Design Suite 14.1

If using a later software version, there may be minor differences between the images and results shown in this document with what you will see in the Design Suite.

UG743 (v14.1) May 8, 2012





Notice of Disclaimer

The information disclosed to you hereunder (the "Materials") is provided solely for the selection and use of Xilinx products. To the maximum extent permitted by applicable law: (1) Materials are made available "AS IS" and with all faults, Xilinx hereby DISCLAIMS ALL WARRANTIES AND CONDITIONS, EXPRESS, IMPLIED, OR STATUTORY, INCLUDING BUT NOT LIMITED TO WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, OR FITNESS FOR ANY PARTICULAR PURPOSE; and (2) Xilinx shall not be liable (whether in contract or tort, including negligence, or under any other theory of liability) for any loss or damage of any kind or nature related to, arising under, or in connection with, the Materials (including your use of the Materials), including for any direct, indirect, special, incidental, or consequential loss or damage (including loss of data, profits, goodwill, or any type of loss or damage suffered as a result of any action brought by a third party) even if such damage or loss was reasonably foreseeable or Xilinx had been advised of the possibility of the same. Xilinx assumes no obligation to correct any errors contained in the Materials or to notify you of updates to the Materials or to product specifications. You may not reproduce, modify, distribute, or publicly display the Materials without prior written consent. Certain products are subject to the terms and conditions of the Limited Warranties which can be viewed at <http://www.xilinx.com/warranty.htm>; IP cores may be subject to warranty and support terms contained in a license issued to you by Xilinx. Xilinx products are not designed or intended to be fail-safe or for use in any application requiring fail-safe performance; you assume sole risk and liability for use of Xilinx products in Critical Applications: <http://www.xilinx.com/warranty.htm#critapps>.

©Copyright 2012 Xilinx, Inc. Xilinx, the Xilinx logo, Artix, ISE, Kintex, Spartan, Virtex, Zynq, 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..

Table of Contents

Software Requirements.....	4
Hardware Requirements.....	4
Tutorial Design Files.....	5
Tutorial Design Description.....	5
Tutorial Software Overview.....	6
Software Tools Flow.....	6
Project Directory and HDL Design Structure.....	7
Step 1: Synthesizing Netlists from HDL Source (Optional).....	9
Step 2: Creating a Project.....	10
Step 3: Creating Reconfigurable Partitions and Adding Reconfigurable Modules.....	12
Step 4: Adding Additional Reconfigurable Modules.....	15
Step 5: Adding Black Box Modules (Optional).....	18
Step 6: Floorplanning Reconfigurable Partitions.....	20
Step 7: Partition Pins and Reconfigurable partition Interface Timing.....	25
Step 8: Running Partial Reconfiguration Design Rule Checks.....	28
Step 9: Implementing and Promoting a Configuration.....	30
Step 10: Creating and Implementing Additional Configurations.....	34
Step 11: Verifying Configurations.....	38
Step 12: Generating and Downloading BIT Files.....	39
Conclusion.....	42

Partial Reconfiguration Tutorial

This tutorial demonstrates how to create a simple partial reconfiguration (PR) design from Hardware Description Language (HDL) synthesis through BIT file generation and download. Xilinx[®] software tools are used to implement and analyze the design through the PlanAhead[™] software. Other tools, such as CORE Generator[™] and ChipScope[™] Pro, can be used with a partial reconfiguration design, but are not described in this tutorial.

To benefit from this tutorial, you must be familiar with partial reconfiguration, and have experience implementing an FPGA design with Xilinx software.

This tutorial covers only a subset of the features in the PlanAhead tool. Additional features are covered in other tutorials.

After completing this tutorial, you will be able to set up, run, and manage a partial reconfiguration project using the PlanAhead tool. Specifically, you will learn how to:

- Create a reconfigurable partition (RP).
- Add a reconfigurable module (RM).
- Define Pblock ranges for the reconfigurable partitions.
- Run pr-specific DRC checks.
- Create and implement configurations.
- Verify configuration.
- Generate bit files.

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/xilinx14_1/iil.pdf.

Hardware Requirements

Xilinx recommends a minimum of 2 GB of RAM when using the PlanAhead tool on larger devices. Although 1 GB is sufficient, it can impact performance.

Tutorial Design Files

This tutorial uses a reference design, `UG743_design_files.zip`, which must be unzipped to a write-accessible directory. You can download a copy of the reference design from http://www.xilinx.com/support/documentation/dt_planahead_planahead14-1_tutorials.htm.

The unzipped 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 data is required each time you run the tutorial.

This tutorial includes a project file that has already been implemented. To reduce the data size, some implementation files were removed from the design, leaving only the required results data in the run directories.

Tutorial Design Description

The FPGA design in this tutorial is targeted to the Xilinx ML605 prototype board described at <http://www.xilinx.com/ml605>. The design targets a Virtex®-6 xc6vlx240tff1156-**1** device. The FPGA device drives the LEDs in particular sequences depending on which reconfigurable modules are loaded.

The design contains two reconfigurable partitions:

- One reconfigurable partition contains embedded block RAM.

Reconfiguring the block RAM module with different block RAM data changes the LED sequence of the eight GPIO LEDs.

- One reconfigurable partition contains just general fabric (Slice) logic.

Reconfiguring the this module with different state machine transitions changes the direction that the four LEDs rotate, either clockwise or counterclockwise.

Tutorial Software Overview

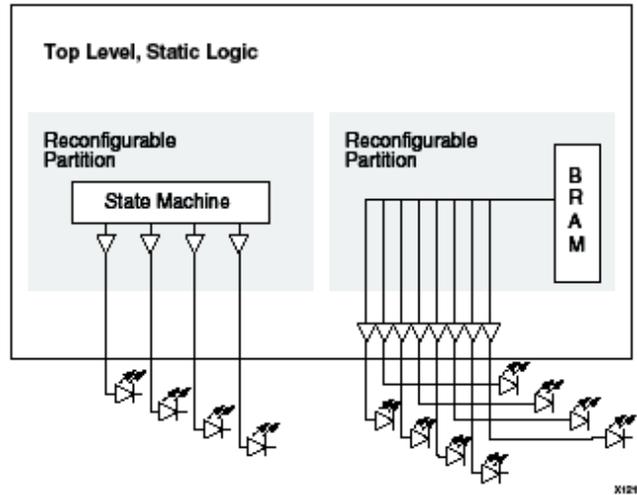


Figure 1: Design Overview - Partial Reconfiguration

Software Tools Flow

Partial reconfiguration uses a bottom-up synthesis approach with top-down implementation methodology. This tutorial uses the Xilinx Synthesis Technology (XST) to synthesize the design, and the PlanAhead tool to implement the design. Other tools and methodologies can be used to successfully implement a partial reconfiguration design.

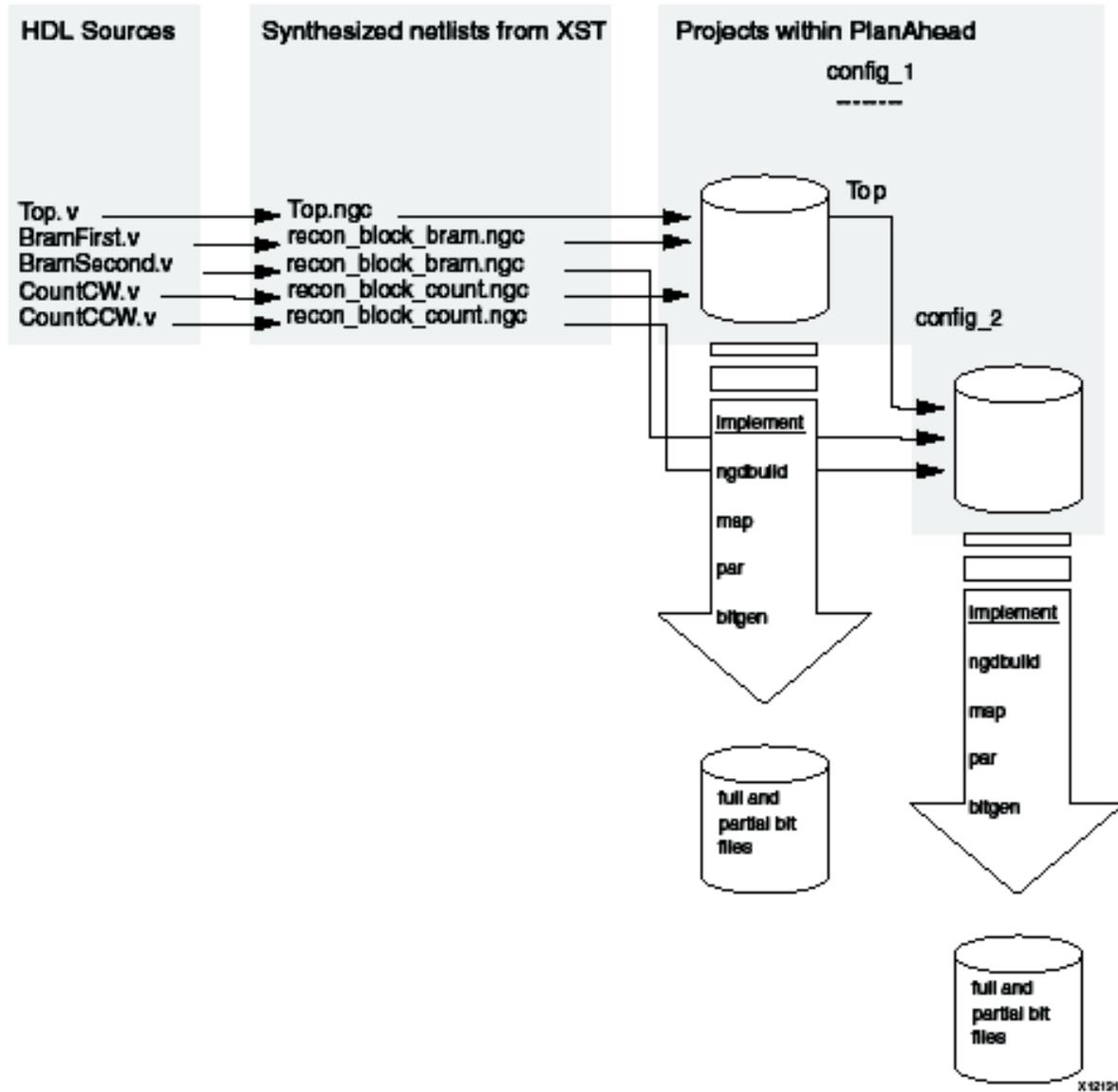


Figure 2: Software Flow Overview

Project Directory and HDL Design Structure

A black box and bottom-up synthesis approach is required to correctly structure and synthesize a partially reconfigurable FPGA design. Each reconfigurable module is synthesized as an individual project generating its own netlist. Since the top-level design instantiates the reconfigurable modules as black boxes, the reconfigurable module netlists are not included in the top-level netlist.

The directory structure of the tutorial design is:

```

<Extract_Dir>
- Implementation      -> Synthesis results (and implementation
                        results if scripted methodology used)
- Synth               -> PlanAhead project and results
- PlanAhead           -> PlanAhead project and results
- Source              -> HDL source files and constraint files
- Tools               -> Tcl scripts for command line flow (not
                        Covered in this tutorial)
    
```

Because each reconfigurable module is synthesized independently, there is a directory for the top-level module, as well as each reconfigurable module in the /Source and /Synth directories.

```

Implementation
Top (top level and all static logic)
BramFirst (first version of the BRAM RM)
BramSecond (second version of the BRAM RM)
CountCW (Clockwise version of the counter)
CountCCW (Counterclockwise version of the counter)
    
```

Open the top-level source file:

```
<Extract_Dir>/Source/Top/Top.v
```

The two reconfigurable modules in this design (recon_block_bram and recon_block_count) are declared as black boxes in the HDL. No underlying HDL descriptions are provided for these blocks.

Step 1: Synthesizing Netlists from HDL Source (Optional)

Since the PlanAhead tool does not support HDL projects for the partial reconfiguration flow, you must synthesize the design using XST before creating a PlanAhead project. XST has already been run in the files accompanying this tutorial. This allows you to use the accompanying NGC files if you wish. To use the accompanying NGC files, skip to “Step 2: Creating a Project”.

The XST project files are:

- `<Extract_Dir>/Synth/Top/Top.xst`
- `<Extract_Dir>/Synth/Top/Top.prj`

In `Top.xst`, automatic I/O buffer insertion is turned on by default. However, reconfigurable modules cannot have I/O buffers and this option must be set to **NO** in the XST project files for all reconfigurable modules.

-iobuf NO

Running the Tcl Scripts to Synthesize All Modules

To synthesize all modules, run the following command from the `<Extract_Dir>` directory.

```
tclsh ./Tools/xpartition.tcl ./Tools/data_synth.tcl
```

This script calls XST to synthesize the Verilog files in the source directories.

The NGC netlist files generated by XST are stored in the `<Extract_Dir>/Synth/<module>` directories.

If you prefer to use Synplify Pro, modify the `data_synth.tcl` file before running the above as follows:

**SYNTH_TOOL “synplify_pro” **

The Synplify Pro project files provided in the `<Extract_Dir>/Synth/<module>` directories are used when this option is set.

Step 2: Creating a Project

To create a new project:

1. Open the PlanAhead tool.
 - In Windows, select **Start > All Programs > Xilinx Design Tools > Xilinx ISE Design Suite 14.1 > PlanAhead > PlanAhead**.
 - In Linux, change the directory to `<Extract_Dir>/planAhead`, and enter **planAhead**.
2. On the PlanAhead Getting Started page, select **Create New Project**.
3. In the Create a New PlanAhead Project dialog box, click **Next**.
4. In the Project Name dialog box, set the project name and select the project location to be:
`<Extract_Dir>/planAhead`
5. Click **Next**.
6. In the Design Source dialog box, select **Specify Synthesized (EDIF or NGC) Netlist**.
7. Check the **Enable Partial Reconfiguration** box.
8. Click **Next**.

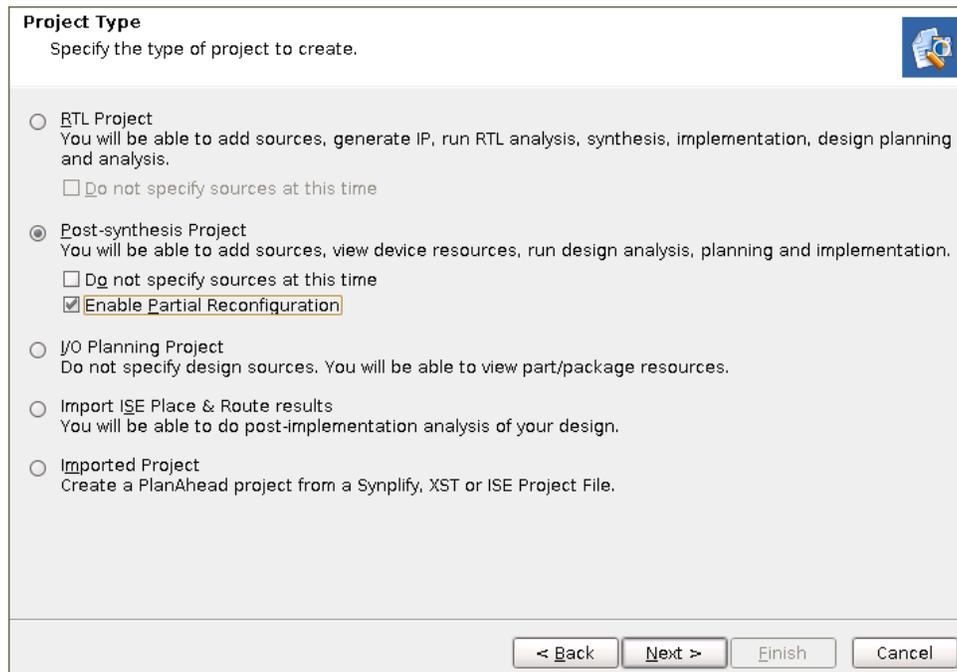


Figure 3: Specify Design Source

9. Click the **Add Files** button and browse to the following top-level netlist:
`<Extract_Dir>/Synth/Top/Top.ngc`
10. Click **OK** to add this file to the project.

Only files associated with the top-level (or Static) logic should be at this point. Netlists for the lower level, reconfigurable modules will be added later.

11. Click **Next**.
12. On the Constraint Files page, click **Add Files**.
13. Browse to the following User Constraints File (UCF):

`<Extract_Dir>/Source/UCF/top_m1605.ucf`

14. Click **OK**.
15. Click **Next**.

The Default Part wizard scans the netlist and picks the appropriate device.

16. Verify that the selected device is **xc6vlx240tff1156-1**.
17. Click **Next**.
18. On the New Project Summary page, verify the project settings.

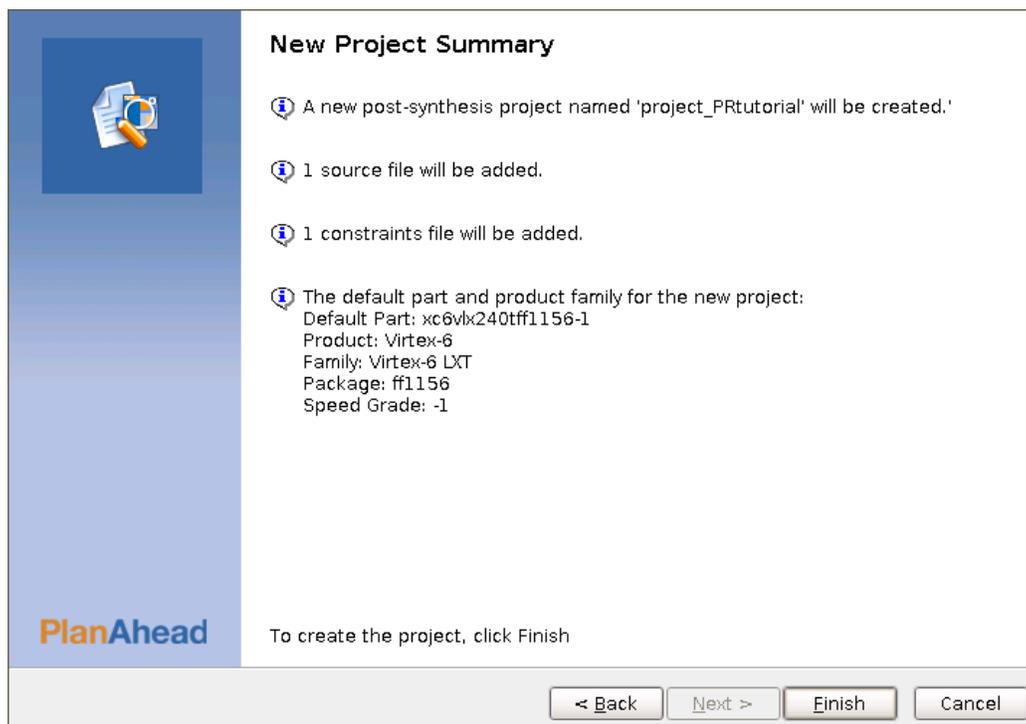


Figure 4: New Project Summary

19. Click **Finish**.

Step 3: Creating Reconfigurable Partitions and Adding Reconfigurable Modules

This step includes:

- Creating a Reconfigurable Partition for U1_RP_Bram
- Creating a Reconfigurable Partition for U2_RP_Count

Creating a Reconfigurable Partition for U1_RP_Bram

Many of the features described in this tutorial are available only when the Synthesized Design is open. If you close the Synthesized Design, or if you close and reopen the project, click **Open Synthesized Design** in the Flow Navigator to open the Synthesized Design.

To create a reconfigurable partition for **U1_RP_Bram**:

1. In the Flow Navigator, select **Open Synthesized Design**.

This loads the netlist and constraints into memory. Because you have not yet assigned any netlists to the reconfigurable modules, the software displays a message about undefined instances.

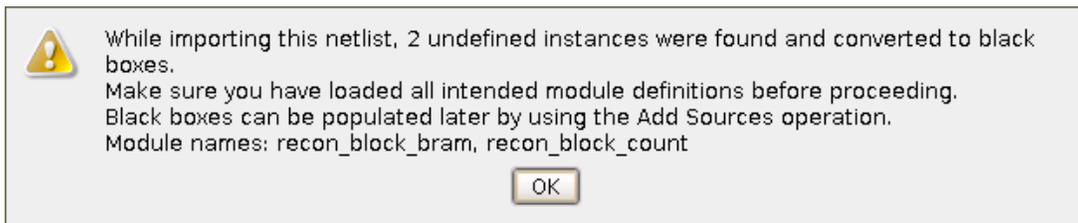


Figure 5: Undefined Modules Warning

2. Click **OK** to accept the message about the black box modules.
3. In the Netlist window, select **U1_RP_Bram**.
4. Right-click.
5. Select **Set Partition**.

This launches the Set Partition wizard.

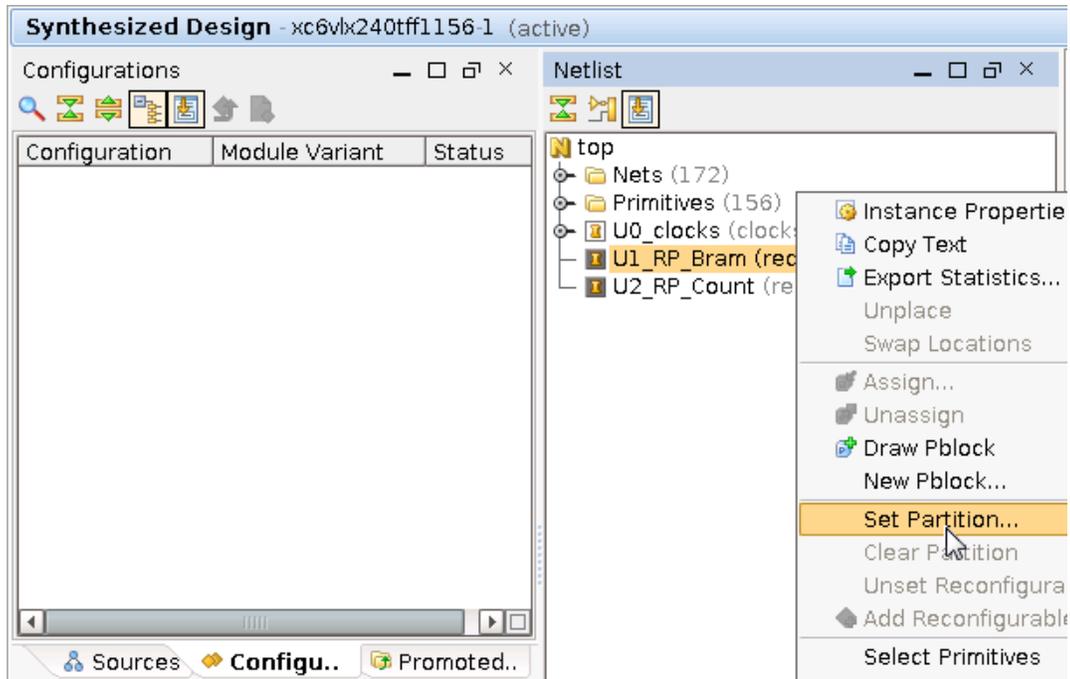


Figure 6: Setting Partition and Reconfigurable Module

6. Click **Next**.
7. Select **is a reconfigurable Partition** if not already selected.
8. Click **Next**.
9. Name the reconfigurable module **BramFirst**.
10. Select **Netlist already available for this Reconfigurable Module** if not already selected.
11. Click **Next**.
12. For the **Top Netlist File**, browse to and select:


```
<Extract_Dir>/Synth/BramFirst/recon_block_bram.ngc
```
13. Click **OK**.

You can add optional netlist directories here if the reconfigurable module has lower-level netlists associated with it. In this case, there are none.
14. Click **Next**.
15. Click **Next** to bypass the optional constraint files screen.

You can add module-level constraint files here. In this case, there are none.
16. Verify the **Set Partition Summary**.
17. Click **Finish**.

A reconfigurable partition has now been created for **U1_RP_Bram**, and there is one reconfiguration module listed under this instance as shown the following figure. A Pblock has also been created for this instance, and can be seen in the Physical Constraints window.

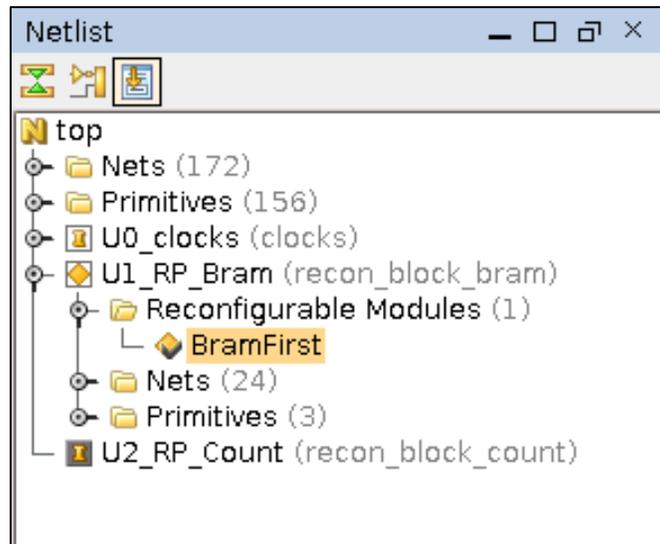


Figure 7: Reconfigurable Module "BramFirst"

Creating a Reconfigurable Partition for U2_RP_Count

To create a reconfigurable partition for **U2_RP_Count**:

1. Select **U2_RP_Count** from the Netlist window.
2. Right-click.
3. Select **Set Partition**.
4. Select **is a reconfigurable Partition** if not already set.
5. Name the reconfigurable module **CountCW**.
6. Select **Netlist already available for this Reconfigurable Module** if not already set.
7. Set **Top Netlist File** to:


```
<Extract_Dir>/Synth/CountCW/recon_block_count.ngc
```
8. Click **Next** until the Summary page is presented.
9. Click **Finish** to complete the wizard.

There is now one reconfigurable module for each reconfigurable partition. "Step 4: Adding Additional Reconfigurable Modules" describes how to add additional reconfigurable modules for a reconfigurable partition.

Step 4: Adding Additional Reconfigurable Modules

This step includes:

- Adding a Reconfigurable Module to U1_RP_Bram
- Adding a Reconfigurable Module to U2_RP_Count

Note that adding a new reconfigurable module is almost identical to creating the reconfigurable partition and adding the first reconfigurable module.

Adding a Reconfigurable Module to U1_RP_Bram

To add a reconfigurable module to **U1_RP_Bram**:

1. Go to the Netlist window.
2. Select **U1_RP_Bram**.
3. Right-click.
4. Select **Add Reconfigurable Module**.

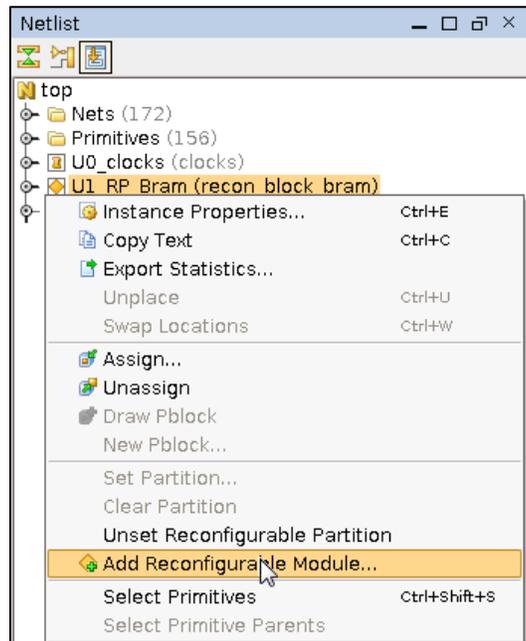


Figure 8: Adding a Reconfigurable Module

5. Click **Next** on the Add Reconfigurable Module wizard.
6. Name the new reconfigurable module **BramSecond**.
7. Click **Next**.
8. Set **Top Netlist File** to:

`<Extract_Dir>/Synth/BramSecond/recon_block_bram.ngc`

9. Click **Next** until the summary page is presented.
10. Click **Finish** to complete the wizard.

Adding a Reconfigurable Module to U2_RP_Count

To add a reconfigurable module to **U2_RP_Count**:

1. Select **U2_RP_Count**.
2. Right-click.
3. Select **Add Reconfigurable Module**.
4. Name the new reconfigurable module **CountCCW**.
5. Set **Top Netlist File** to:

`<Extract_Dir>/Synth/CountCCW/recon_block_count.ngc`

6. Complete the wizard.

One additional reconfigurable module has now been added to each reconfigurable partition. The netlist window should resemble the figure below.

Nets and primitives specific to the currently-active reconfigurable module are listed under each reconfigurable partition. They are signified by a yellow diamond with a check mark.

In the figure below, the active modules are **BramFirst** and **CountCW**.

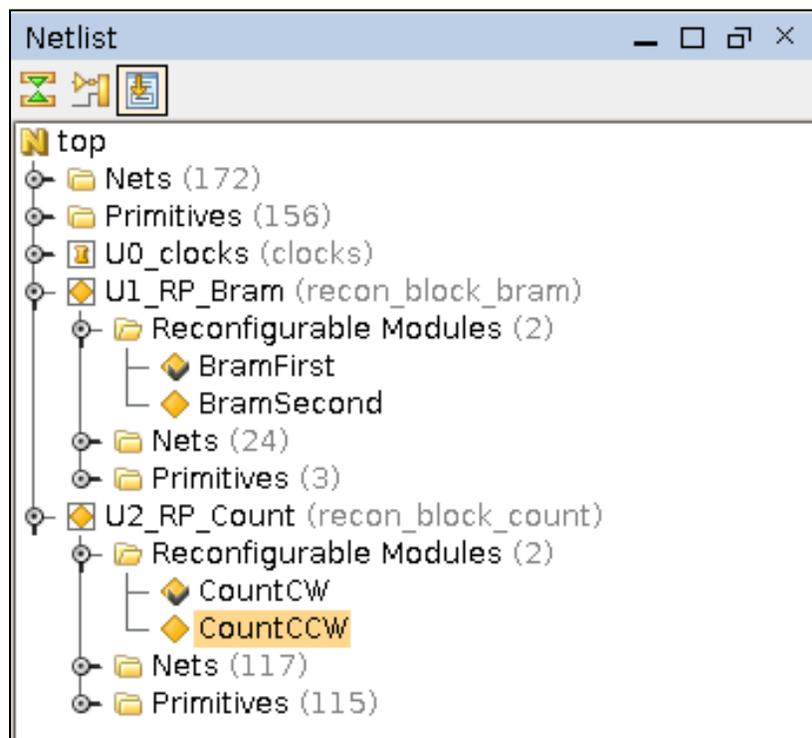


Figure 9: Netlist Window After Adding Reconfigurable Modules

Changing the Active Reconfigurable Module

To change the active reconfigurable module to a different reconfigurable module:

1. Select a reconfigurable module.
2. Right-click.
3. Select **Set as Active Reconfigurable Module**.

Primitives Associated with Reconfigurable Modules

The resources for each reconfigurable module can be seen under the Primitives folder in the Netlist window. Depending on which RM is active, the resources will change. The following resources are used by the current active RMs.

- A RAMB36 and an inverter for the reconfigurable module associated with U1_RP_Bram
- Slice logic (LUT, XORY, and FDR) for the module associated with U2_RP_Count

The RAMB36 primitive must be included in the Area Group Range constraints for the associated reconfigurable partition. This is shown in "Step 6: Floorplanning Reconfigurable Partitions".

Step 5: Adding Black Box Modules (Optional)

The purpose of creating a black box module is to generate a blanking BIT file in the BitGen step.

This step includes:

- Adding a Black Box Module to U1_RP_Bram
- Adding a Black Box Module to U2_RP_Count

Adding a Black Box Module to U1_RP_Bram

To add a black box module to **U1_RP_Bram**:

1. Select **U1_RP_Bram**.
2. Right-click.
3. Select **Add Reconfigurable Module**.
4. Name the new reconfigurable module **BramBB**.
5. Select **Add this Reconfigurable Module as a black box without a netlist**.
6. Click **Next**.

Because there is no netlist or constraint file associated with a black box module, the wizard does not prompt you for more information.

7. Click **Finish**.

Adding a Black Box Module to U2_RP_Count

To add a black box module to **U2_RP_Count**:

1. In the Netlist window, select **U2_RP_Count**.
2. Right-click.
3. Select **Add Reconfigurable Module**.
4. Set **Reconfigurable Module Name** to **CountBB**.
5. Select **Add this Reconfigurable Module as a black box without a netlist**.
6. Finish the wizard.

One black box module now appears under each reconfigurable partition in the netlist window as shown in the figure below.

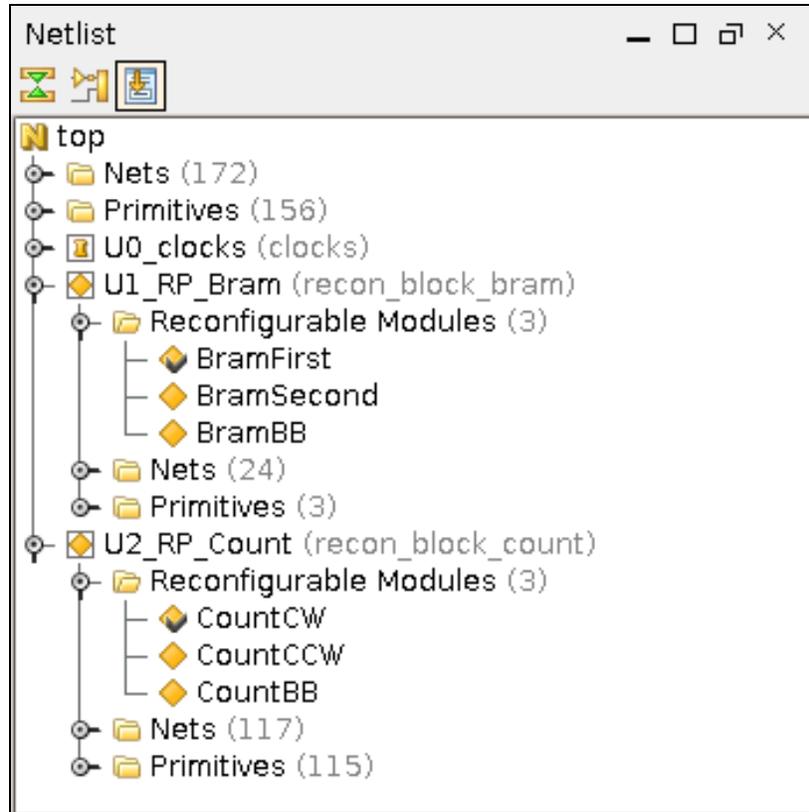


Figure 10: Netlist Window with Black Box Modules

Step 6: Floorplanning Reconfigurable Partitions

Each reconfigurable partition must have an AREA_GROUP range constraint to designate which physical resources are part of the reconfigurable partition. In this example, the reconfigurable partitions are **U1_RP_Bram** and **U2_RP_Count**.

All physical resources not part of the AREA_GROUP range constraint associated with a reconfigurable partition are part of the static logic. Static logic is unaffected by partial reconfiguration, and remains operational during reconfiguration. Refer to the Partial Reconfiguration User's Guide for more information on the affects of active logic during partial reconfiguration, and the use of decoupling logic.

Do not create the AREA_GROUP range constraints until the reconfigurable partitions have been created with the **Set Partition** command as described in "Step 3: Creating Reconfigurable Partitions and Adding Reconfigurable Modules".

Creating the AREA_GROUP Range for pblock_U1_RP_Bram

To create the AREA_GROUP range for **pblock_U1_RP_Bram**:

1. In the Netlist window, verify that **BramFirst** and **CountCW** are the active modules.
Active modules are indicated by a check mark next to the module name.
2. If **BramFirst** and **CountCW** are not the active modules:
 - a) Select **BramFirst** or **CountCW**.
 - b) Right-click.
 - c) Select **Set As Active Reconfigurable Module**.

The PlanAhead tool reports the required resources for the AREA_GROUP range for the active reconfigurable module. Be sure that a black box reconfigurable module is not active.

For situations in which different reconfigurable modules associated with a reconfigurable partition use various resources, the AREA_GROUP range of the reconfigurable partition must contain a superset of resources used by all its reconfigurable modules.

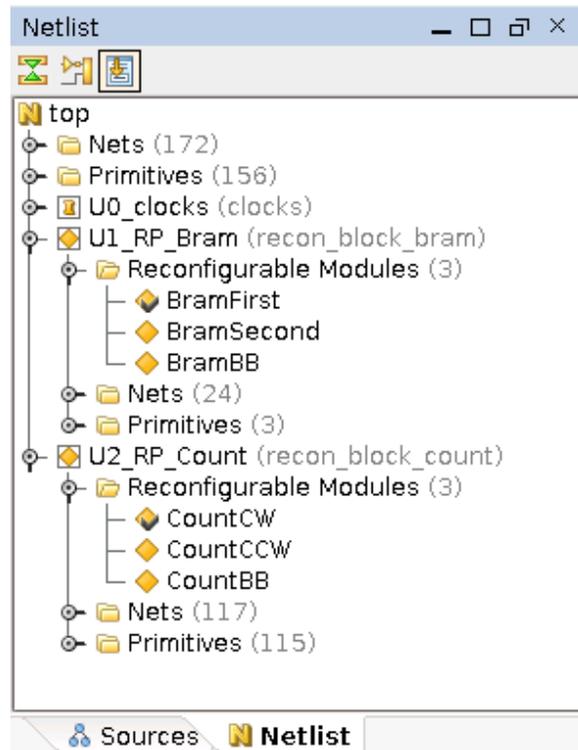


Figure 11: Setting Active Reconfiguration Modules

3. Click **Layout** on the toolbar.
4. Select **Floorplanning** to change the current view from Design Analysis to Floorplanning. The Physical Constraints window shows a list of all the current Pblocks. In the PlanAhead tool, AREA_GROUP constraints are called Pblocks. Pblocks are automatically created for any modules defined as a reconfigurable partition.
5. In the Physical Constraints window, select the Pblock named **pblock_U1_RP_Bram**.
6. Right-click and select **Set Pblock Size** .
7. Draw a box that encompasses some slice logic and at least one RAMB36. Note that as you drag the mouse to make the Pblock rectangle, the tool tip will tell you the resources being selected.
8. After drawing the box, select both SLICE and RAMB36 resources for the AREA_GROUP.

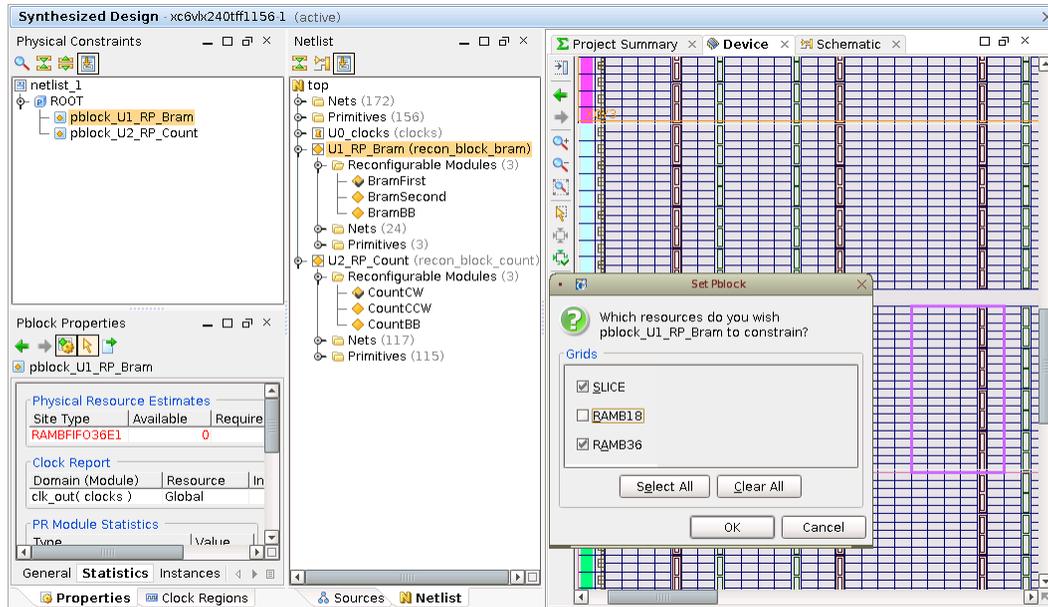


Figure 12: Setting Pblock Resources for pblock_U1_RP_Bram

9. Click **OK**.

Creating an AREA_GROUP Range for pblock_U2_RP_Count

To create an AREA_GROUP range for **pblock_U2_RP_Count**:

1. Select **pblock_U2_RP_Count**.
2. Use the Set Pblock Size tool to draw a box that encompasses slice resources.
3. After drawing the rectangle, select **SLICE** resources for the AREA_GROUP.

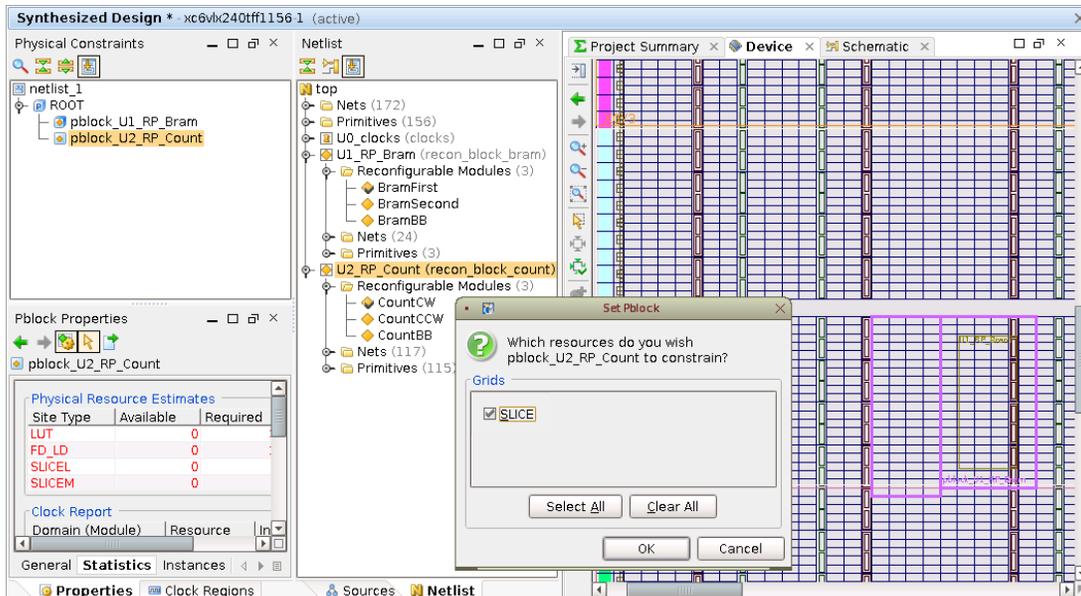


Figure 13: Setting Pblock Resources for pblock_U2_RP_Count

4. Click **OK**.
5. Review the floorplan and make any necessary adjustments.

The location of the of Pblocks is not critical as long as the correct resources are selected. The rules for placement of the Pblocks are:

- The Pblocks may not overlap
- The Pblocks may not share the same configuration frames. To avoid this place the Pblocks side by side instead of on top/bottom of each other.

The resulting floorplan may resemble the following figure.

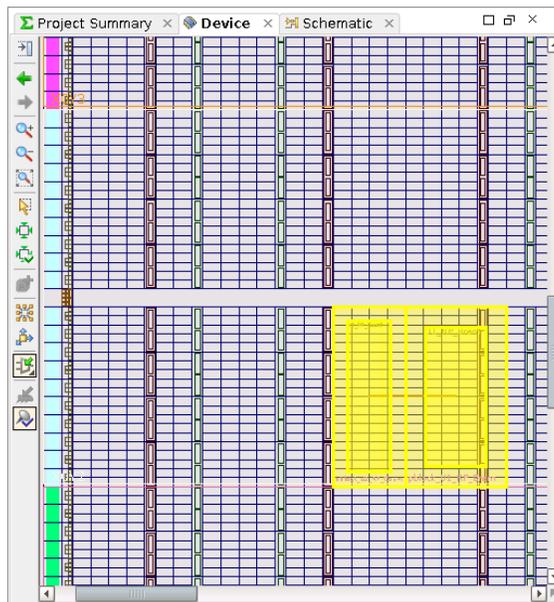


Figure 14: Final AREA_GROUP Ranges

6. Verify that the RAMB36 required for **U1_RP_Bram** is included in the AREA_GROUP range.
 - a) In the Physical Constraints window, select **pblock_U1_RP_Bram**.
 - b) Click the **Statistics** tab.

In this instance, the available number of RAMBFIFO36E1 exceeds the required amount.

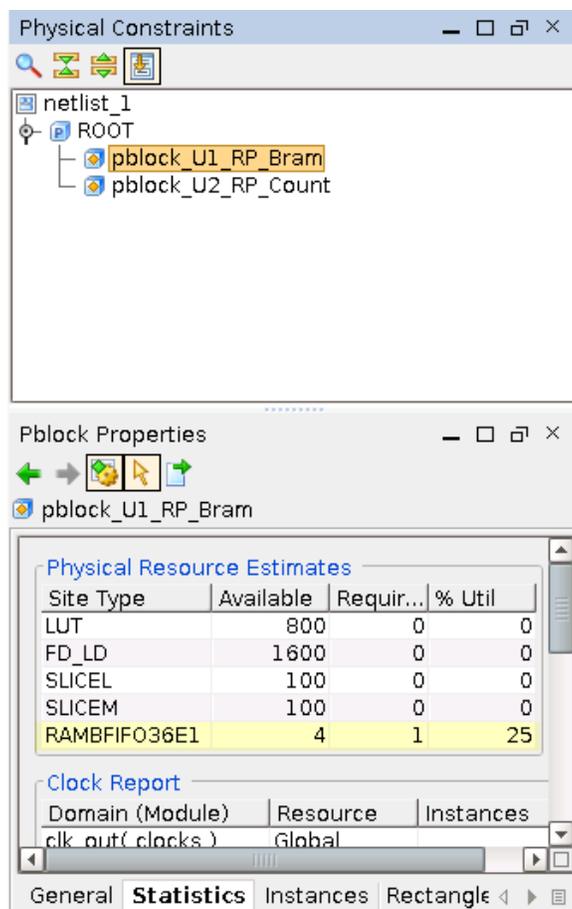


Figure 15: Pblock Statistics -- pblock_U1_RP_Bram

7. Select **File > Save Design**.

8. In the Flow Navigator, click **Project Manager**.

The Sources window opens showing all the source and constraint files.

9. Under Constraints, double-click the UCF file.

The AREA_GROUP constraints are similar to the following.

```
INST "U1_RP_Bram" AREA_GROUP = "pblock_U1_RP_Bram";
AREA_GROUP "pblock_U1_RP_Bram" RANGE=SLICE_X32Y80:SLICE_X41Y99;
AREA_GROUP "pblock_U1_RP_Bram" RANGE=RAMB36_X2Y16:RAMB36_X2Y19;
INST "U2_RP_Count" AREA_GROUP = "pblock_U2_RP_Count";
AREA_GROUP "pblock_U2_RP_Count" RANGE=SLICE_X24Y80:SLICE_X31Y99;
```

10. Return to the Synthesized Design view.

11. Click **Layout**.

12. Select **Default Layout** to revert to the default view in the PlanAhead tool.

Step 7: Partition Pins and Reconfigurable partition Interface Timing

Partition pins are required in the partial reconfiguration flow for all reconfigurable partition interface signals that are not global logic or dedicated routes. Partition pins provide a known routing connection to the reconfigurable partition. They are automatically inserted by NGDBuild when implementation is run.

The current implementation of partition pins requires proxy logic (a LUT1). A LUT1 is inserted in the input and output paths of the reconfigurable partition. Xilinx recommends that you register these inputs and outputs on both sides of the reconfigurable partition boundary. Registering these inputs helps minimize any timing closure issues related to the reconfigurable partition interface.

If these guidelines are followed, it is likely that a simple PERIOD constraint is enough to constrain this interface. However, in situations with tight timing requirements, you might need to:

- Create TPSYNC constraints on the partition pin, or
- Add LOC constraints to the static logic to minimize routing delays between the static logic and the partition pin.

Because the same static logic implementation is used for every configuration, try to meet timing with the most timing critical reconfigurable modules first. The reconfigurable partition interfaces in this tutorial are very similar between the reconfigurable modules. A global PERIOD constraint is sufficient to meet reconfigurable partition interface timing.

Adding a PERIOD Constraint

To add a PERIOD constraint:

1. Open the Timing Constraints window (**Window > Timing Constraints**) in the Synthesized Design window.
2. In the Timing Constraints window, right-click and select **Create Timing Constraint**.
3. In the Create a New Timing Constraint dialog, under the Basic group (TNM) category, set the following values:
 - Group name
clk_p
 - Group type
Net
 - TNM type
TNM_NET

- Predefined group
Leave blank
 - Net
clk_p
4. Click **OK**.
 5. Re-open the Create Timing Constraint tool.
 6. Under the TimeSpec period category, set the following values:
 - TimeSpec name
TS_clk_p
 - Period
5 ns
 7. Click the **Browse** icon () next to Group.
 8. Choose **User Defined** in the Group constraints type field.
 9. Set **clk_p** in the User defined groups field, and click **OK**.
 10. Click **OK** again to create the constraint.

The Timing Constraint window now looks like the following figure.

This global timing constraint constrains all synchronous paths connected to **clk_p**. This timing constraint is not specific to the reconfigurable partition interface.

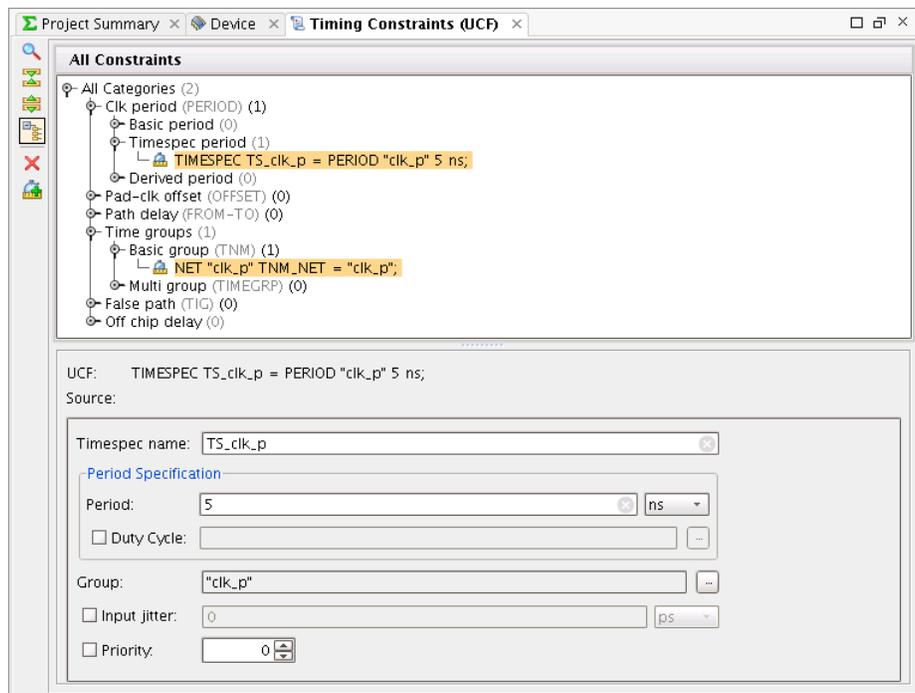


Figure 16: Timing Constraints View

11. Save the design.
12. Navigate back to the Project Manager to view the UCF.

The following constraints are now included in the constraint file.

```
TIMESPEC TS_clk_p = PERIOD "clk_p" 5 ns;  
NET "clk_p" TNM_NET = "clk_p";
```

If the UCF is still open from previous steps, select **Reload** in the yellow banner to reload the modified file.

Step 8: Running Partial Reconfiguration Design Rule Checks

You must follow many design rules specific to partial reconfiguration in order to implement a valid design. Some of these rules have been incorporated in the Design Rule Check (DRC) engine under the Partial Reconfig and Partition headings.

Run these checks on the partial reconfiguration design before implementing configurations and generating BIT files.

On a typical design, you might want to run all PlanAhead DRCs. This tutorial runs only the **Partial Reconfig** and **Partition** DRC.

Running the Partial Reconfiguration and Partition DRCs

To run the partial reconfiguration and partition DRCs:

1. From the Flow Navigator, select **Report DRC**, located under Synthesized Design.

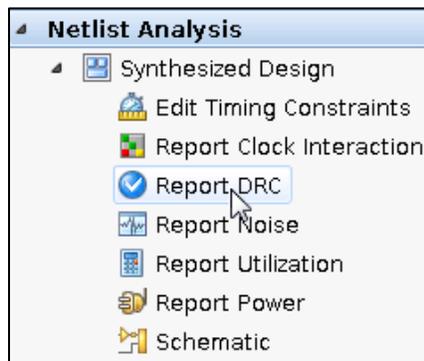


Figure 17: Report DRC

2. From the Report DRC dialog box, select the **Partition and Partial Reconfig** rules and click **OK** to run the selected DRC rules.

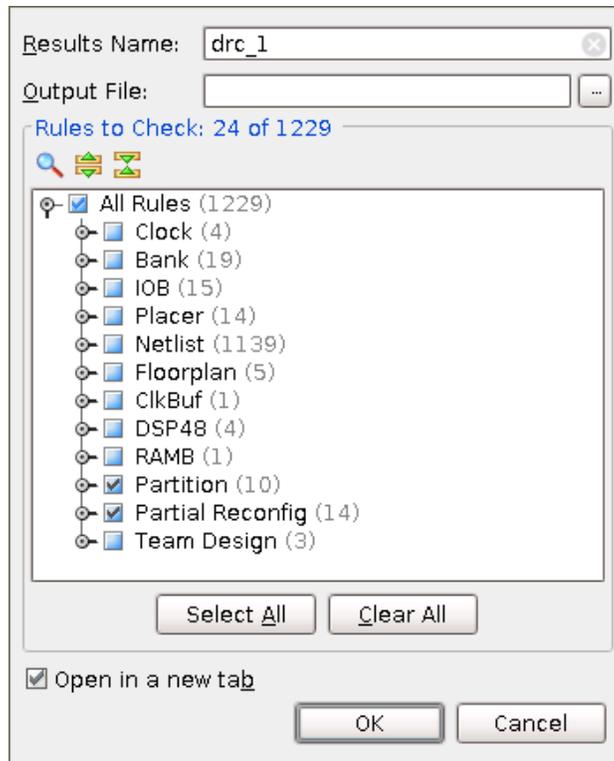


Figure 18: Partition and Partial Reconfig DRCs

3. Review the Severity of the messages returned by the DRC. The messages can have a Severity of:
 - Advisory
 - Warning
 - Error
 - Fatal

In this case, the worst Severity returned is a Warning, and can be safely ignored for this tutorial.

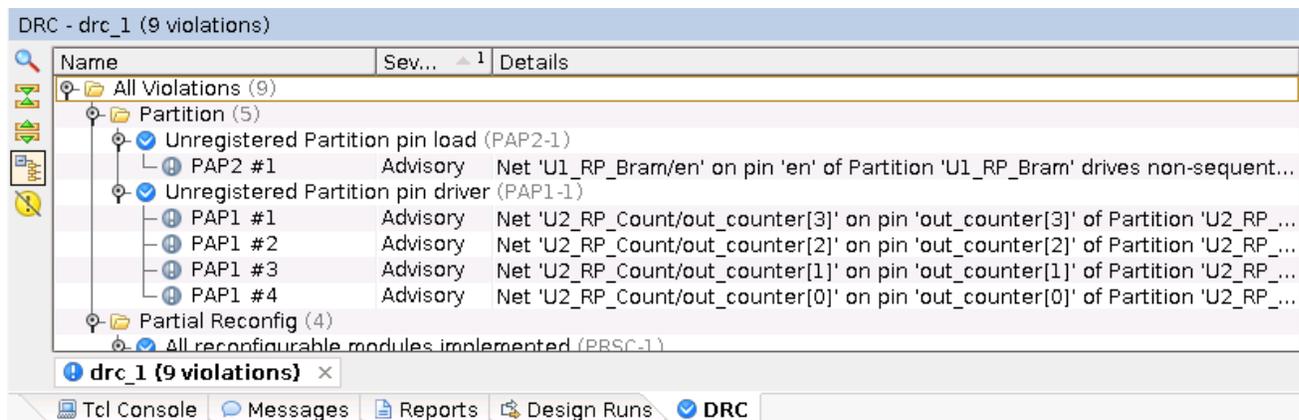


Figure 19: DRC Results

Step 9: Implementing and Promoting a Configuration

Although each reconfigurable partition can have multiple reconfigurable modules associated with it, only one reconfigurable module can be implemented at any one time for the reconfigurable partition.

The set of active reconfigurable modules along with static logic is called a *configuration* and is a complete design. Multiple configurations exist for a partial reconfiguration project to allow different permutations of reconfigurable modules to be implemented, thus generating full and partial BIT files. Each configuration is its own independent implementation run.

The resulting output files are:

- Format files
- NGD
- NGM
- NCD
- PCF
- Report files

You can run Xilinx software tools and debug techniques on each configuration individually, such as opening an NCD with FPGA Editor or performing a gate-level simulation.

The design in this tutorial can be fully implemented with only two configurations using these reconfigurable module sets and resulting BIT files:

```
Configuration
-----
config_1 RMs: BramFirst, CountCW
Bits: config_1.bit (full bit file)
config_1_U1_RP_Bram_BramFirst_partial.bit
config_1_U2_RP_Count_CountCW_partial.bit

config_2 RMs: BramSecond, CountCCW
Bits: config_2.bit (full bit file)
config_2_U1_RP_Bram_BramSecond_partial.bit
config_2_U2_RP_Count_CountCCW_partial.bit
```

In summary:

- The full BIT file `config_1.bit` contains reconfigurable modules `BramFirst` and `CountCW`.
- The full BIT file `config_2.bit` contains reconfigurable modules `BramSecond` and `CountCCW`.

Two other configuration sets can generate unique, full BIT files. However, because they reuse modules previously implemented, the partial BIT files are identical to the partial BIT files generated in the configurations above.

```
Configuration
-----
config_3 RMs: BramFirst, CountCCW
Bits: config_3.bit (full bit file)
config_3_U1_RP_Bram_BramFirst_partial.bit
config_3_U2_RP_Count_CountCCW_partial.bit

config_4 RMs: BramSecond, CountCW
Bits: config_4.bit (full bit file)
config_4_U1_RP_Bram_BramSecond_partial.bit
config_4_U2_RP_Count_CountCW_partial.bit
```

Finally, if blackbox RMs were created, one additional configuration can be implemented to generate the blanking bitstreams. The full bit file will contain no logic for blackbox RPs, and the partial bit files can be used to blank an RP.

```
Configuration
-----
config_5 RMs: BramBB, CountBB
Bits: config_3.bit (full bit file)
config_3_U1_RP_Bram_BramBB_partial.bit
config_3_U2_RP_Count_CountBB_partial.bit
```

The implementation of a configuration in the PlanAhead tool is called a *run*. A run must be created for each configuration.

The PlanAhead tool creates a configuration when the project is created. The reconfigurable modules set for this configuration depend on the order in which the reconfigurable modules were added to the project. The first reconfigurable module defined for each reconfigurable partition is set for this configuration (**BramFirst** and **CountCW** in the case of this tutorial).

An FPGA device configured with a full BIT file contains the reconfigurable modules that were implemented in the configuration. If the system requires that only the static logic is functioning after loading the full BIT file, implement a configuration containing black boxes for all reconfigurable partitions. The resulting partial BIT files are essentially blank files for the reconfigurable partitions.

Implementing Configuration config_1

To implement configuration **config_1**:

1. Verify that the reconfigurable modules set for **config_1** are **BramFirst** and **CountCW**.
 - a) In the Design Runs window (**Window > Design Runs** if not already open), select **config_1**.
 - b) In the Implementation Run Properties window, select the **Partitions** tab. See the figure below.
 - c) Ensure that the Module Variants listed are **BramFirst** and **CountCW**.

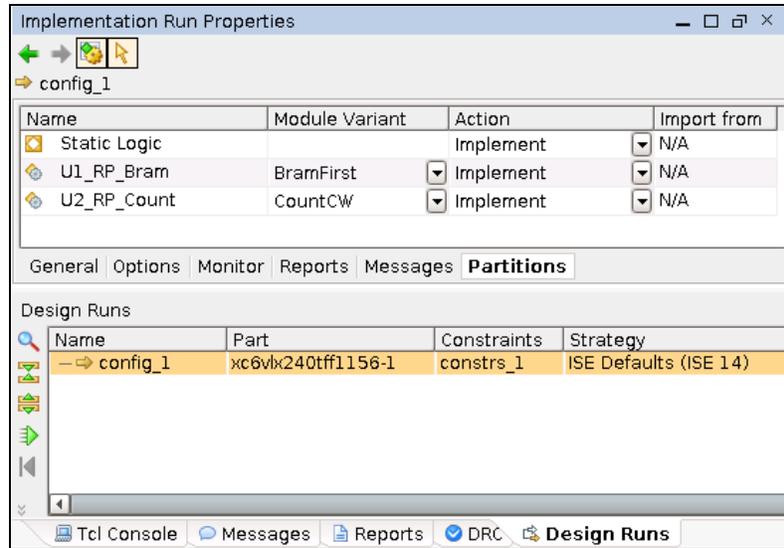


Figure 20: Configuration Module Variants -- config_1

2. In the Flow Navigator, click **Run Implementation** to launch the implementation run. Click Save if prompted.

The Design Runs window Status field shows when NGDBuild, Map, PAR, and TRACE are running. This can also be monitored in the status bar in the upper right-hand corner of the PlanAhead tool main window. For more details about the run status, see the Compilation Log window.

3. The Implementation Completed dialog box appears after implementation is complete. There are several options here, but for this tutorial just click **Cancel**.
4. Promote the **config_1** configuration. From the Flow Navigator (or the Implementation Complete dialog), click **Promote Partitions**.

Now that **config_1** has been successfully implemented, it can be promoted. Running other configurations without first promoting (then importing) one configuration results in incompatible partial BIT files among configurations.

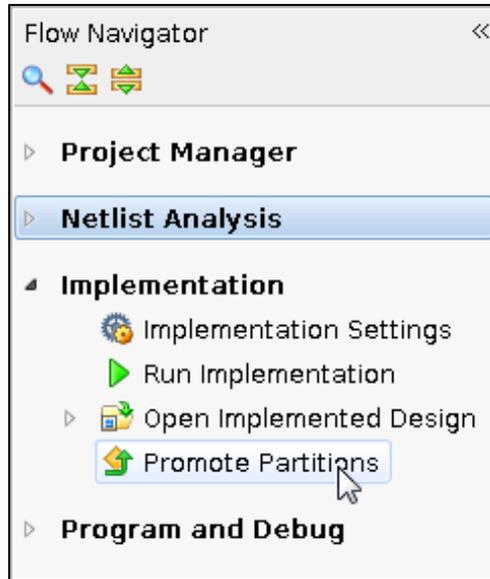


Figure 21: Selecting the Promote Partitions Button

- In the Promote Partitions dialog box, make sure the **Automatically manage Partition action and import location** option is checked, and click **OK**.
- Select the **Configurations** tab. (Select **Window > Configurations** if not already open.) The Status is listed as Promoted.

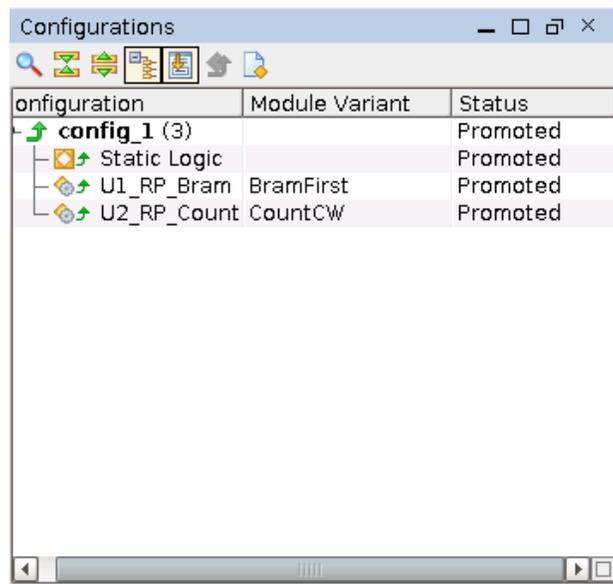


Figure 22: Verifying the Promoted Status

Step 10: Creating and Implementing Additional Configurations

To create a new configuration:

1. From the Flow Navigator, right-click on Implementation and select **Create New Implementation Runs**.

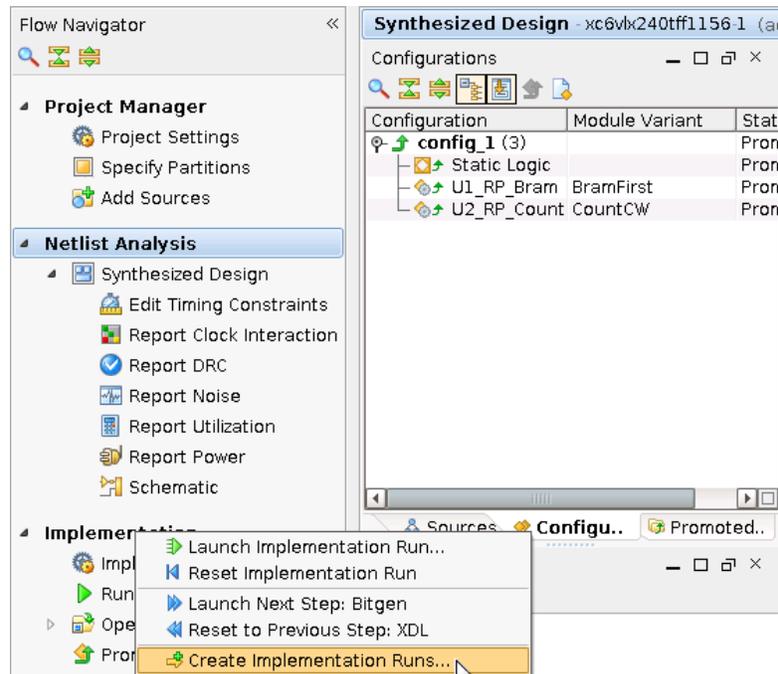


Figure 23: Creating New Implementation Runs

2. In the introduction page of the Create Multiple Runs wizard, click **Next**.
3. In the Set-Up Implementation Run page, click **Next**.

In the Choose Implementation Strategies and Reconfigurable Modules page, you can create multiple configurations, choose implementation strategies, and define which reconfigurable modules make up the configurations. There is already a new configuration listed called **config_2**.

4. In the Choose Implementation Strategies and Reconfigurable Modules page, leave the new configuration name as **config_2**.
5. Click the button in the Partition Action column to open the Specify Partition dialog box.

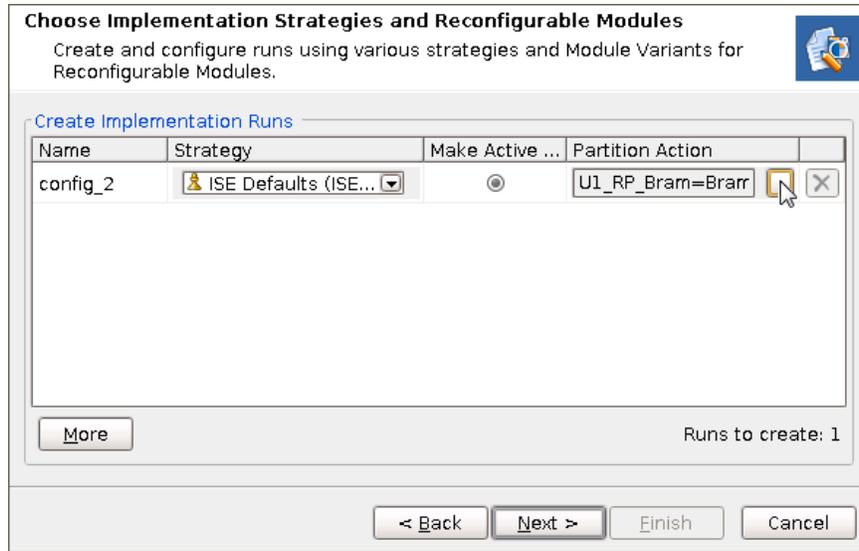


Figure 24: Create Implementation Runs

The default Module Variant is based on the reconfigurable modules currently active in the project (in this case, **BramFirst** and **CountCW**). Because these modules have already been implemented and imported, they are set to **Import** and have an import location set.

6. To create a configuration that implements **BramSecond** and **CountCCW**, change the Module Variant column to match these reconfigurable modules. See the figure below.

Because these reconfigurable modules have not been implemented (or promoted), the Action field changes to **Implement**.

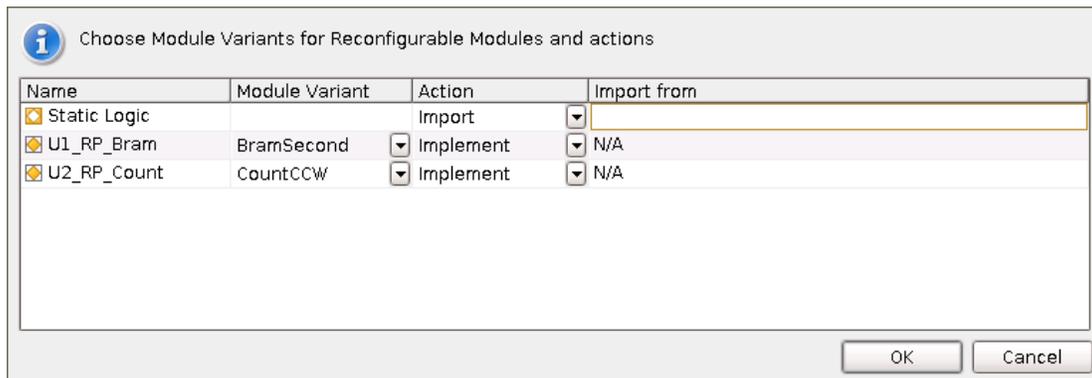


Figure 25: Specify Partition Dialog Box -- BramSecond/CountCCW

7. In the Specify Partition dialog box, click **OK**.
8. Optional. If you optionally created Black Box module variants, create an additional configuration to implement those modules.
 - a) Click **More** in the Choose Implementation Strategies and Reconfigurable Modules page to add **config_3**.
 - b) Open the **Specify Partition** dialog box.

- c) Set the Module Variants to **BramBB** and **CountBB**.
- d) Click **OK**.

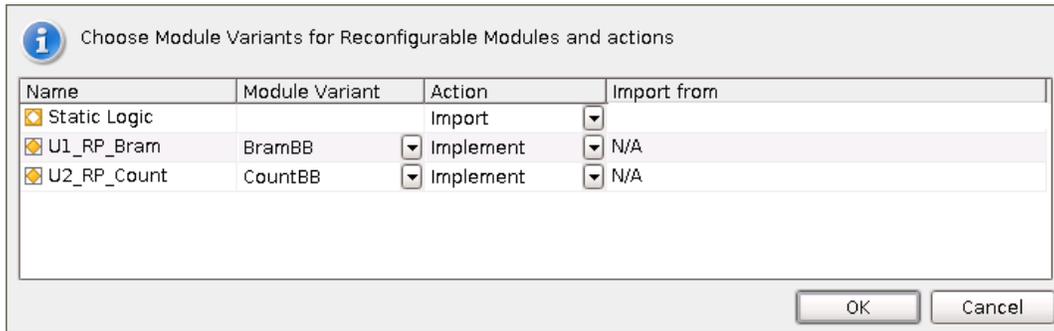


Figure 26: Specify Partition Dialog Box -- BramBB/CountBB

- 9. On the Create Implementation Run page, click **Next**.
- 10. On the Launch Options page, select **Do Not Launch Now** and click **Next**.
- 11. Click **Finish** on the Create New Runs Summary page.

The Design Runs window displays the new configurations that were created through the wizard.

- 12. Select a new configuration in the Design Runs window.
- 13. In the Implementation Run Properties window, click the **Partitions** tab to verify the Module Variants and Action fields. The Static logic is set to **Import**, and will be imported from the promoted results from **config_1**.

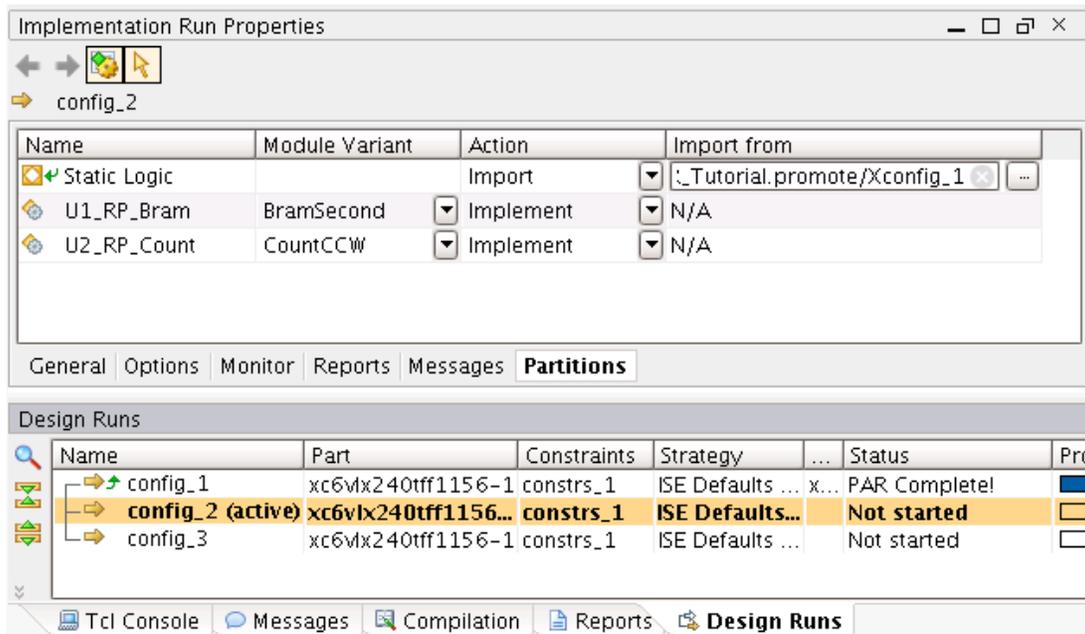


Figure 27: Verifying Configuration Settings

- 14. To launch a configuration:

- a) Select one or more configuration in the **Design Runs** window.
- b) Right click and select **Launch Runs**.

You can launch multiple new configurations in parallel (on multiple processors, if available). The results are not dependent on each other, but are dependent only on the Static logic from **config_1**.

15. In the Launch Selected Runs dialog box:

- a) Select **Launch Runs On Local Host**.
- b) Select the number of jobs (number of processors to use).
- c) Click **OK**.

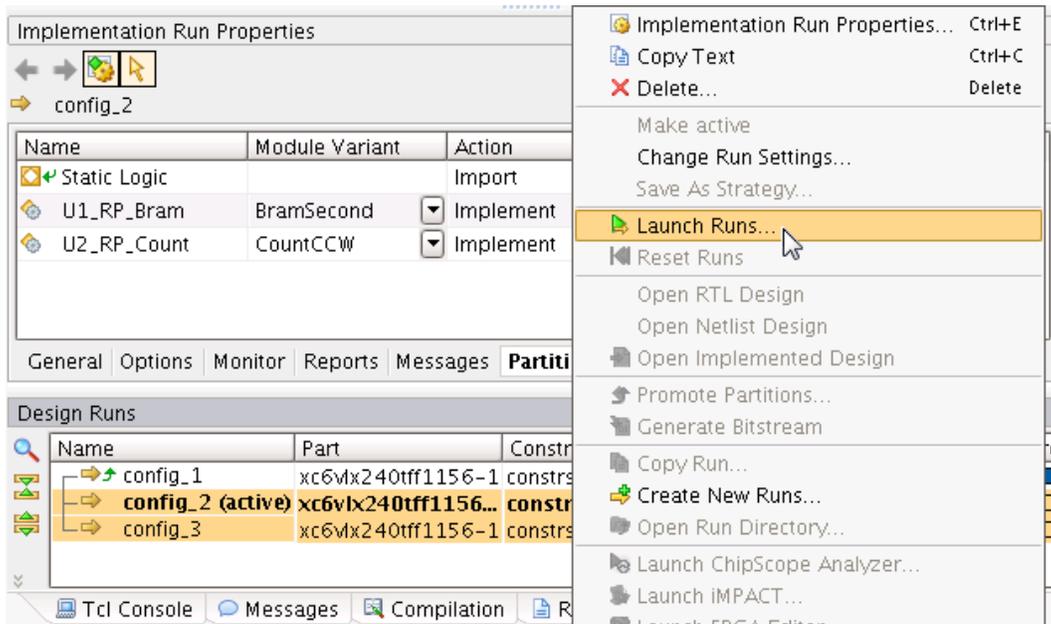


Figure 28: Launching Runs

Step 11: Verifying Configurations

After multiple configurations are implemented, you can compare them to verify that the static logic and partitions pins are consistent across all configurations. Run **Verify Configuration** to ensure the BIT files are compatible.

Running Verify Configuration on All Configurations

To run **Verify Configuration** on all configurations:

1. From the Flow Navigator, select **Verify Configuration** under Program and Debug.

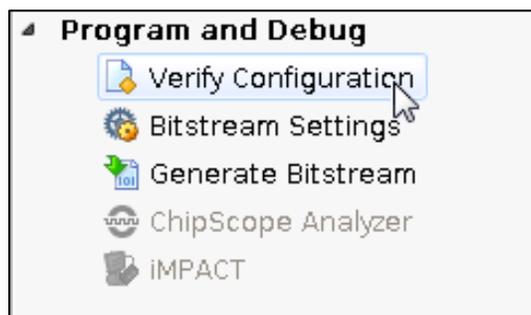


Figure 29: Verifying Configuration

2. Shift-click or Ctrl-click two or more configurations to verify against each other.
3. Note the value shown for **Verification Output File**.

Modify this value if necessary as this is the log file for **pr_verify**. If **Verify Configuration** fails, this file will be useful for analyzing the results.

4. Click **OK**.

If the verification checks pass, the PlanAhead tool reports that no errors were found. You may now generate BIT files.

5. Click **OK**.

Step 12: Generating and Downloading BIT Files

For each configuration, multiple BIT files are generated.

- One full BIT file that can be used to program the FPGA initially.
- One partial BIT file for each reconfigurable partition that contains the logic for the module variants associated with the particular configuration. Any of the partial BIT files can be used to reconfigure the associated partial reconfiguration regions.

In this tutorial you created configurations with:

- BramFirst
- CountCW
- BramSecond
- CountCCW
- BramBB (optional)
- CountBB (optional)

Running BitGen on each configuration creates partial BIT files for all reconfigurable modules. Regardless of the full BIT file used to configure the device initially, you can use any of the partial BIT files to reconfigure the associated partial reconfiguration regions.

Generating BIT Files for all Configurations

To generate BIT Files for all configurations:

1. In the Design Runs window:
 - a) Highlight all configurations.
 - b) Right-click.
 - c) Select **Generate Bitstream**.

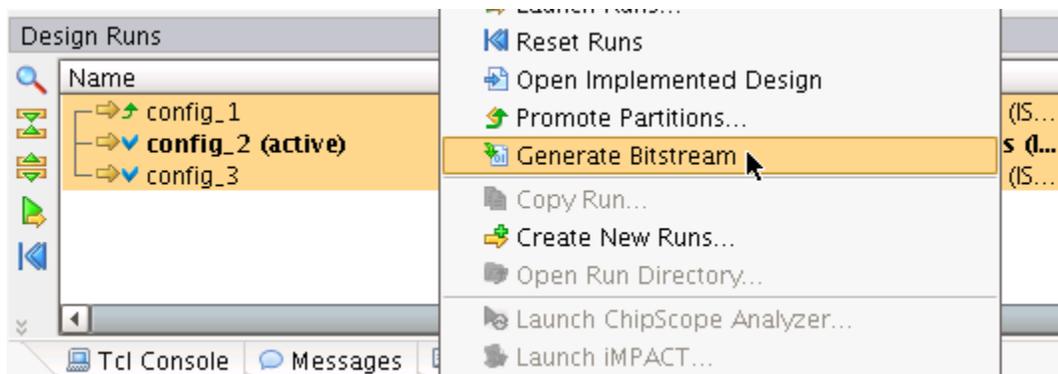


Figure 30: Generate Bitstream

2. No special BitGen options are required to generate partial BIT files. Click **OK** in the BitGen Options dialog box to launch BitGen.

The partial BIT files are downloaded to the FPGA device in the same manner as full BIT files. You can use the iMPACT tool for BIT file downloading, verification, and debugging.

3. Connect the USB download cable to the ML605 board and the PC.
4. To start iMPACT in standalone mode, from the Flow Navigator, select **iMPACT** from the Program and Debug drop-down box.

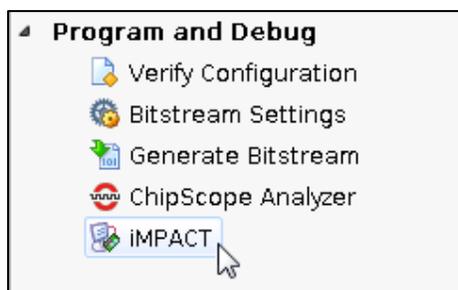


Figure 31: Launching iMPACT

5. In the iMPACT Flows frame:
 - a) Double-click **Boundary Scan**.
 - b) Click **Initialize Chain**.
6. After the chain has been successfully detected:
 - a) Right-click the **xc6vlx240t** device.
 - b) Select **Assign New Configuration File**.
 - c) Select the following full BIT file:

```
<Extract_Dir>/PlanAhead/<project_name>/<project_name>.runs/config_1/  
config_1.bit
```

7. Right-click **xc6vlx240t** again.
8. Select **Program**.

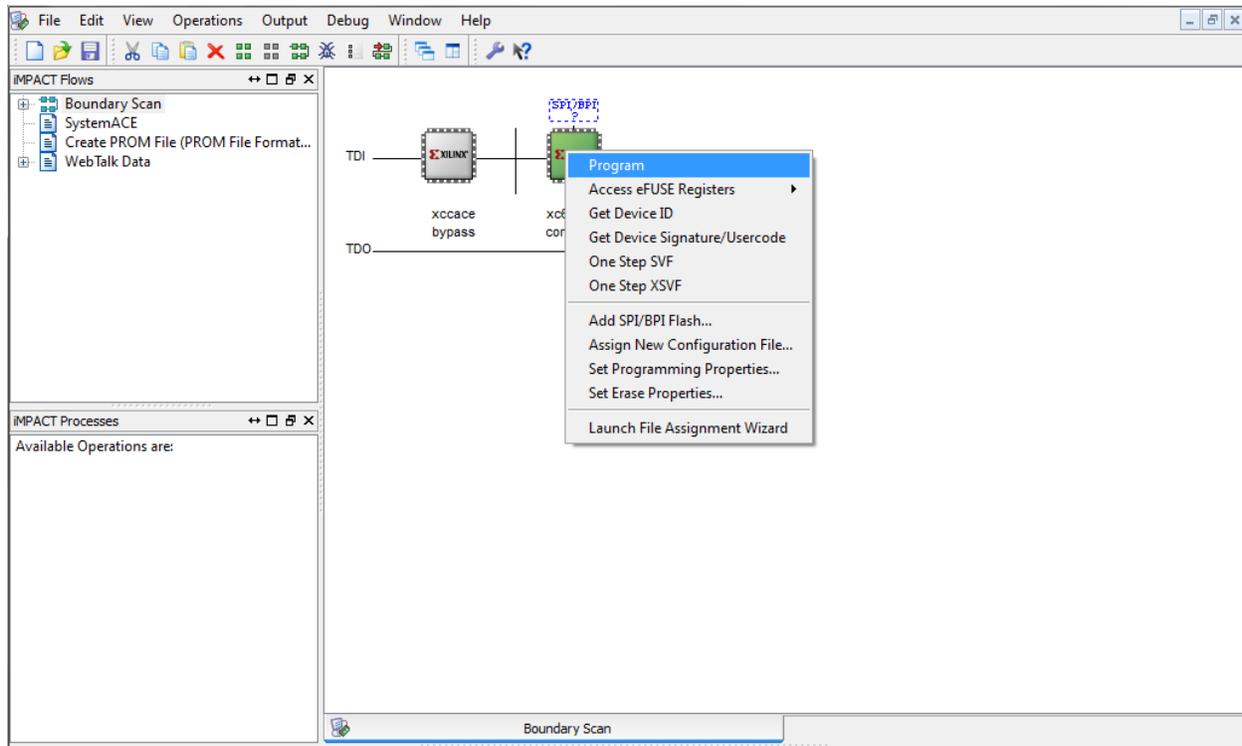


Figure 32: Configuring the Device through iMPACT

The FPGA device on the ML605 programs with the full BIT file.

Associating a Partial BIT File

To associate a partial BIT file:

1. Right-click the **xc6vlx240t** device.
2. Select Assign New Configuration File.
3. Select the following partial BIT file:


```
<Extract_Dir>/PlanAhead/<project_name>/<project_name>.runs/config_2/
config_2_U1_RP_Bram_BramSecond_partial.bit
```
4. Right-click the **xc6vlx240t** device again.
5. Select **Program**.

Conclusion

In this tutorial, you did the following:

- Created a partial reconfiguration project in the PlanAhead tool.
- Created two reconfigurable partitions.
- Associated multiple reconfigurable modules with each reconfigurable partition.
- Constrained each reconfigurable partition to an area of the device using AREA_GROUP constraints.
- Created global timing constraints to constrain the entire design.
- Implemented and promoted an initial configuration.
- Created additional configurations that imported the static logic from the initial configuration.
- Verified the cohesiveness of all configurations.
- Generated BIT files.
- Downloaded a full BIT file to the ML605 board containing the `BramFirst` and `CountCW` modules.
- Reconfigured `U1_RP_Bram` with a partial BIT file for the `BramSecond` module.