PlanAhead Software Tutorial

Overview of the Partial Reconfiguration Flow

UG 743 (v 12.2) July 23, 2010

Demo Design License

The PlanAhead™ software code includes source code for the following programs:

CenterPoint XML

The initial developer of the Original Code is CenterPoint – Connective Software.
Software Engineering GmbH.
Portions created by CenterPoint – Connective Software
Software Engineering GmbH are Copyright © 1998-2000
CenterPoint - Connective Software Engineering GmbH. All Rights Reserved.
Source Code for CenterPoint is available at http://www.cpointc.com/XML/

NLView Schematic Engine

Copyright © Concept Engineering.

Static Timing Engine by Parallax Software Inc.

Copyright © Parallax Software Inc.

Java Two Standard Edition

Includes portions of software from RSA Security, Inc. and some portions licensed from IBM are available at http://oss.software.ibm.com/icu4j/

Powered By JIDE

http://www.jidesoft.com

The BSD License for the JGoodies Looks
Copyright© 2001-2010 JGoodies Karsten Lentzsch. All rights reserved.
Redistribution and use in source and binary forms, with or without modification, are permitted, provided the following conditions are met:

Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

Neither the name of JGoodies Karsten Lentzsch nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Free IP Core License

This is the Entire License for all of our Free IP Cores.

Redistribution and use in source, netlist, binary and silicon forms, with or without modification, are permitted provided that the following conditions are met:

Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

Neither the name of ASICS World Services, the Authors and/or the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS “AS IS” AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Table of Contents

PlanAhead Software Tutorial

Overview of the Partial Reconfiguration Flow

Introduction

This tutorial illustrates how to create a simple Partial Reconfiguration (PR) design from 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 need to have knowledge about Partial Reconfiguration as well as experience implementing an FPGA design with Xilinx software. More information about Partial Reconfiguration is available in the *Partial Reconfiguration User Guide* (UG702).

Note: This tutorial covers a subset of the features contained in the PlanAhead software product bundled with ISE® Design Suite version 12. Additional features are covered in detail in other tutorials.

If you have any questions or comments regarding this tutorial, contact Xilinx Technical Support.

Sample Design Data

This tutorial uses a reference design, `UG743_design_files.zip`, which must be unzipped to a directory on your machine. You can download a copy of the reference design from http://www.xilinx.com/tools/partial-reconfiguration.

You must obtain a FlexLM license for Partial Reconfiguration to access the Partial Reconfiguration features. Contact your Xilinx Field Applications Engineer to obtain a free 30 day license, or go to the Xilinx website at: http://www.xilinx.com/getproduct.

Optionally, an ML605 board and a USB download cable to test in hardware can be used.

Each time you run through the tutorial a new copy of the original sample design data is required.

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

Xilinx ISE and PlanAhead Software

The PlanAhead software is installed with the ISE Design Suite 12.2 software. Before starting the tutorial, ensure that the software is operational and the reference design is unzipped and installed.

For PlanAhead installation instructions and information, refer to the *ISE Design Suite 12: Installation, Licensing, and Release Notes* on the Xilinx website:

http://www.xilinx.com/support/documentation/sw_manuals/xilinx12_2/irn.pdf.

Required Hardware

Xilinx recommends 2 GB or more of RAM for use with PlanAhead on larger devices. For this tutorial a smaller design was used. 1 GB of RAM should be sufficient, but it could impact performance.

PlanAhead Software Documentation and Information

For information about the PlanAhead software, see the following documents, which are available with your software:

- *PlanAhead User Guide* (UG632) - Provides detailed information about the PlanAhead software. http://www.xilinx.com/support/documentation/sw_manuals/xilinx12_2/PlanAhead_UserGuide.pdf
- *Floorplanning Methodology Guide* (UG633) - Provides floorplanning hints. http://www.xilinx.com/support/documentation/sw_manuals/xilinx12_2/Floorplanning_Methodology_Guide.pdf
- *Hierarchical Design Methodology Guide* (UG748) - Provides an overview of the PlanAhead hierarchical design capabilities. http://www.xilinx.com/support/documentation/sw_manuals/xilinx12_2/Hierarchical_Design_Methodology_Guide.pdf

For additional information about PlanAhead, including video demonstrations, go to http://www.xilinx.com/planahead.

For more information about Partial Reconfiguration, refer to the *Partial Reconfiguration User Guide* (UG702) available at http://www.xilinx.com/tools/partial-reconfiguration.

Tutorial Objectives

After completing this tutorial, you will be able to set up, run, and manage a PR project through the PlanAhead software. Specifically, you will know how to create reconfigurable partitions, add reconfigurable modules, define Pblock ranges for the reconfigurable partitions, run PR-specific DRC checks, create and implement configurations, run PR Verify, and generate bit files required for partial reconfiguration in hardware.

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 target device is a Virtex®-6 xc6vlx240tff1156-1.The FPGA device drives the LEDs in particular sequences depending on the reconfigurable modules loaded. The design contains two reconfigurable Partitions, one with embedded block RAM and the other with embedded I/O buffers. Reconfiguring the block RAM module with different block RAM data will change the LED sequence of the eight GPIO LEDs. Reconfiguring the I/O module with different state machine transitions will change the direction that the four LEDs rotate, either clockwise or counterclockwise.

Tutorial Software Overview

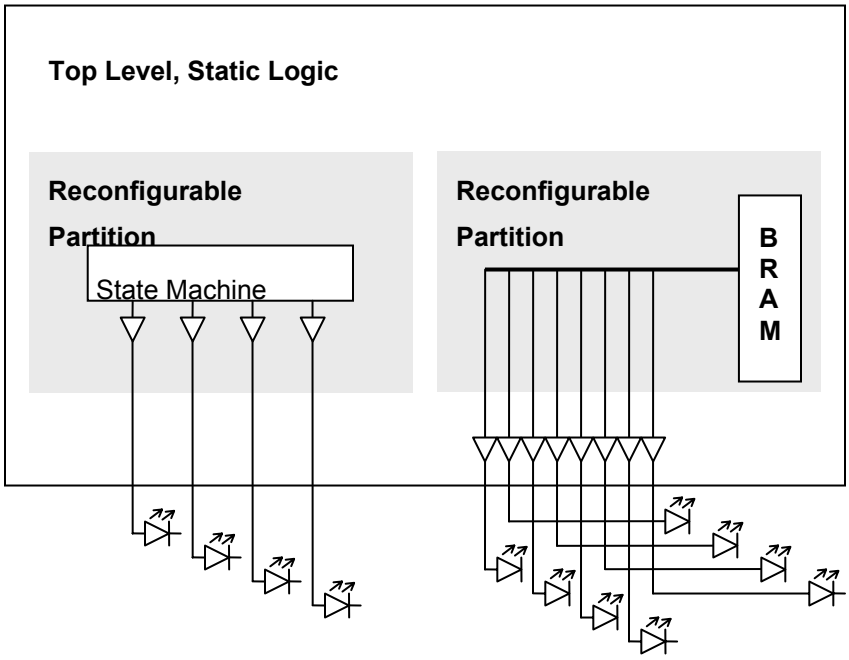

Figure 1: Design Overview – Partial Reconfiguration

Software Tools Flow

Partial Reconfiguration uses a bottom-up synthesis approach with top-down implementation methodology. This particular design and tutorial uses XST to synthesize the design and PlanAhead as the implementation environment. Other tools and methodologies may be used to successfully implement a Partial Reconfiguration design.

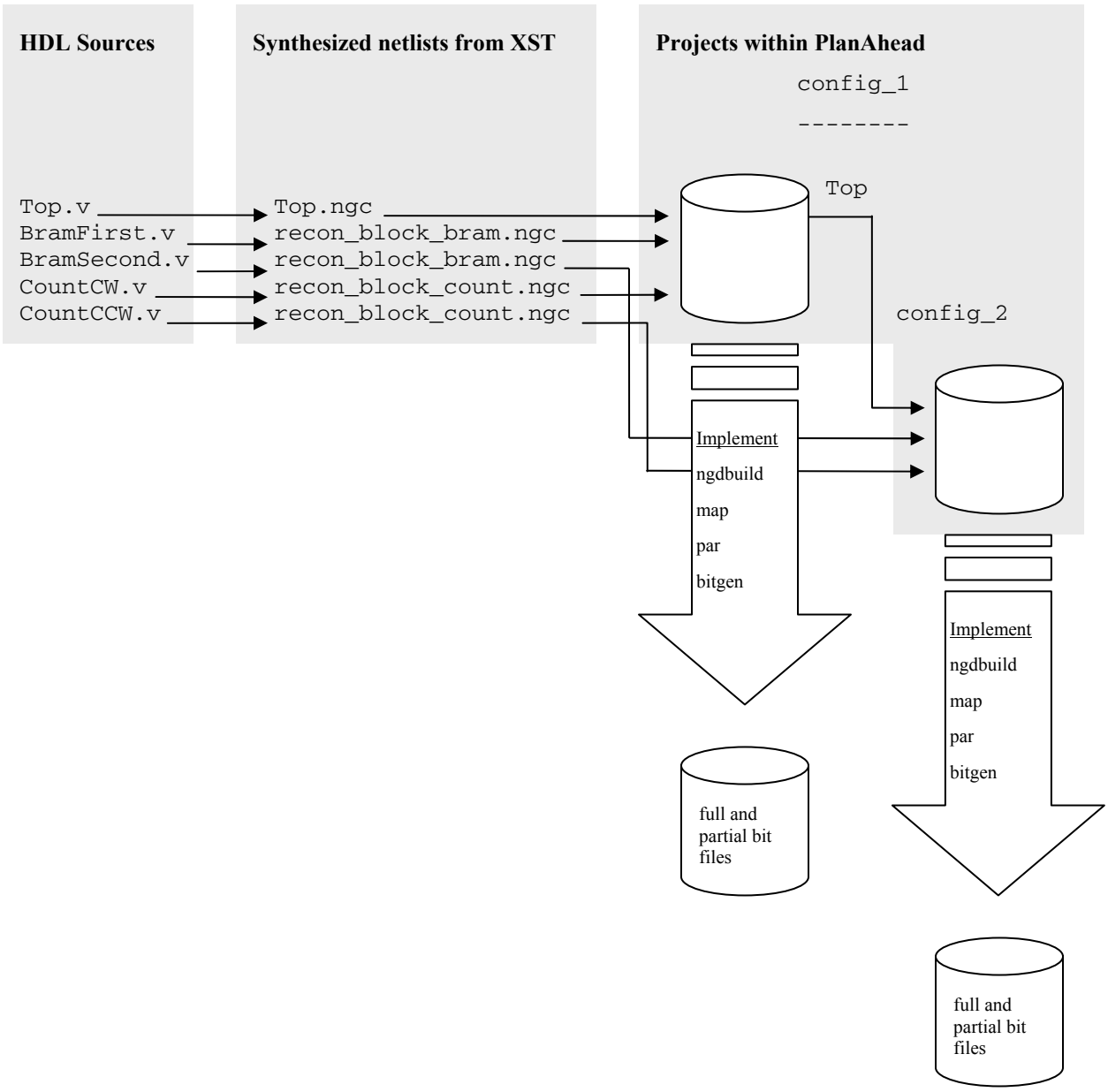

Figure 2: Software Flow Overview

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 (RM) is synthesized as an individual project generating its own netlist. The top-level design instantiates the RMs as black boxes so that the RM netlists are not included in the top-level netlist.

The directory structure of the tutorial design is:

```
xpr_bram_led
  - Implementation → Synthesis results (and implementation
                     results if scripted methodology used)
  - 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 RM is synthesized independently, there is a directory for the top-level module, as well as each RM within the /`Source` and /`Implementation` 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, `xpr_bram_led/Source/Top/Top.v`. Notice that 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.

Tutorial Procedure

This tutorial is separated into steps, followed by general instructions and supplementary detailed steps allowing you to make choices based on your skill level as you progress through the tutorial.

This tutorial has the following steps:

Step 1: Synthesize Netlists from HDL Source (optional as NGCs are provided)

Step 2: Create a PlanAhead Project

Step 3: Create RPs and add RM Modules

Step 4: Add additional RM Modules

Step 5: Add Black Box Modules (optional)

Step 6: Floorplanning Reconfigurable Partitions

Step 7: Partition Pins and RP Interface Timing

Step 8: Partial Reconfiguration Design Rule Checks

Step 9: Implement and Promote a Configuration

Step 10: Create and Implement additional Configurations

Step 11: Run PR Verify

Step 12: Generate and Download Bit Files

If you need help completing a general instruction, go to the detailed steps below it, or if you are ready, skip the step-by-step directions and move on to the next general instruction.

Step 1: Synthesize Netlists from HDL Source (optional) Step 1

The PlanAhead software does not support HDL projects for the Partial Reconfiguration flow. This means that the design must be synthesized using XST prior to creating a PlanAhead project. In the files provided with the tutorial, XST has already been run and the provided NGCs can be used. To take advantage of this and use the NGCs, skip to Step 2.

The XST project files have been created for this design. XST requires two project files to synthesize: `xpr_bram_led/Implementation/Top/Top.xst` and `xpr_bram_led/Implementation/Top/Top.prj`.

In `Top.xst`, notice that automatic I/O buffer insertion is turned on; this is the default.

```
-iobuf YES
```

In the XST project files for all Reconfigurable Modules (RMs), this attribute must be set to `NO` because the lower-level modules cannot have I/O buffers inserted (except in special circumstances which will be described later and illustrated in the U2_RP_Count reconfigurable partition).

1-1. Run the Tcl scripts in the xpr_bram_led/Tools directory to synthesize all the modules.

1-1-1. Run the following command from the `xpr_bram_led/Implementation` directory.

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

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

Note: The `xpr_bram_led/Source/Bram*` modules have block RAM in the RM and `xpr_bram_led/Source/Count*` modules have I/O buffers in the RM. The NGC netlist files generated by XST are stored in the `xpr_bram_led/Implementation/<module>` directories.

1-1-2. If Synplify Pro is the preferred synthesis tool, modify the `data_synth.tcl` file before running the above as follows:

```
SYNTH_TOOL "synplify_pro" \
```

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

Step 2: Open a Project Step 2

2-1. Start PlanAhead and create a new project.

2-1-1. Open the PlanAhead software.

- On Windows, select the Xilinx PlanAhead 12.2 Desktop icon or **Start > Programs > Xilinx ISE Design Suite 12.2 > PlanAhead > PlanAhead**.
- On Linux, change the directory to `<Install_Dir>/PlanAhead_Tutorial/Tutorial_Created_Data`, and enter **`planAhead`**.

The PlanAhead Getting Started Help page opens.

2-1-2. Select the **Create New Project** link.

The Create a New PlanAhead Project confirmation dialog box opens.

2-1-3. Click **Next**.

The Project Name dialog box opens.

2-1-4. Set the project name and project location and click **Next**.

2-1-5. Select **Specify synthesized (EDIF or NGC) netlist**, check the **Set PR Project** box, and click **Next** (Figure 3).

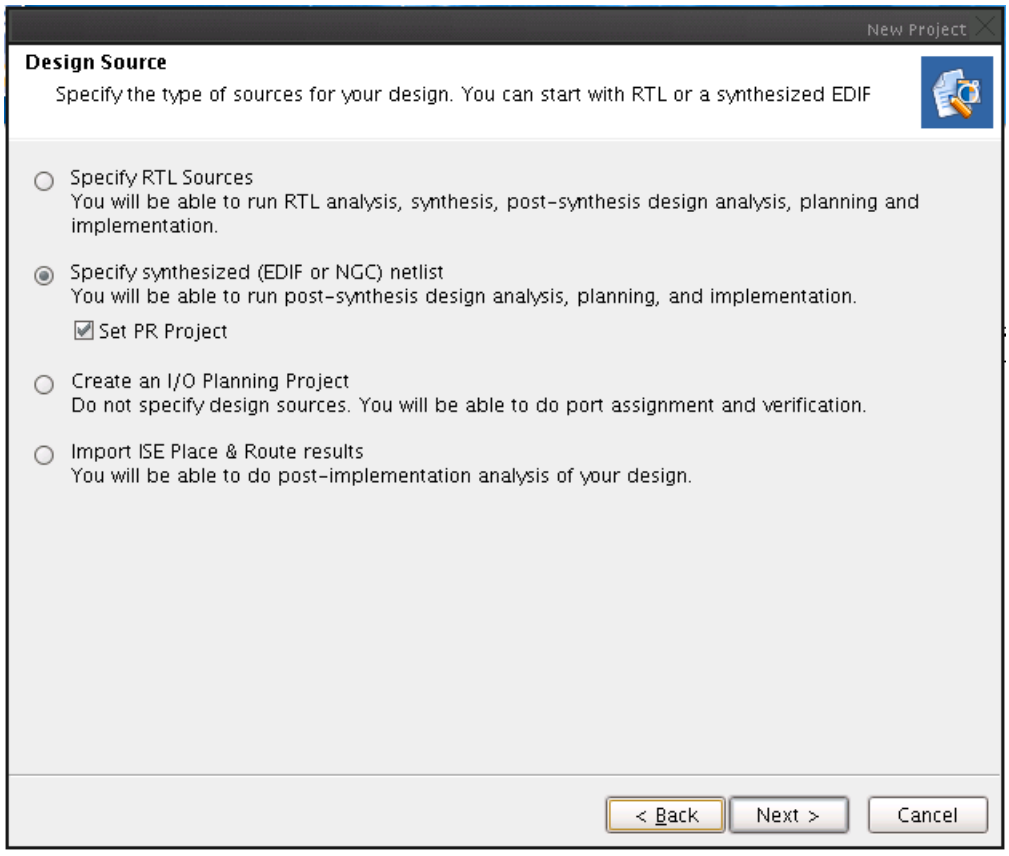

Figure 3: Specify Design Source

2-1-6. Browse to the Top Netlist File at `xpr_bram_led/Implementation/Top/Top.ngc`. Click **Open**, and then click **Next**. Do not set the optional netlist directories.

Note: The optional netlist directories should only be used in a PR project if there are lower-level netlists associated with the static logic. Lower-level netlists associated with Reconfigurable Modules (RMs) will be added later.

2-1-7. On the Constraint Files page, click the **Add Files** button and browse to the user constraints file (UCF) provided at `xpr_bram_led/Source/UCF/top_ml605.ucf`. Click **Save**, and then click **Next**.

2-1-8. The Default Part wizard will scan the netlist and pick the appropriate part. Verify that the selected device is **xc6vlx240tff1156-1**, and click **Next**.

2-1-9. On the New Project Summary page, verify the project settings as shown in Figure 4, and click **Finish**.

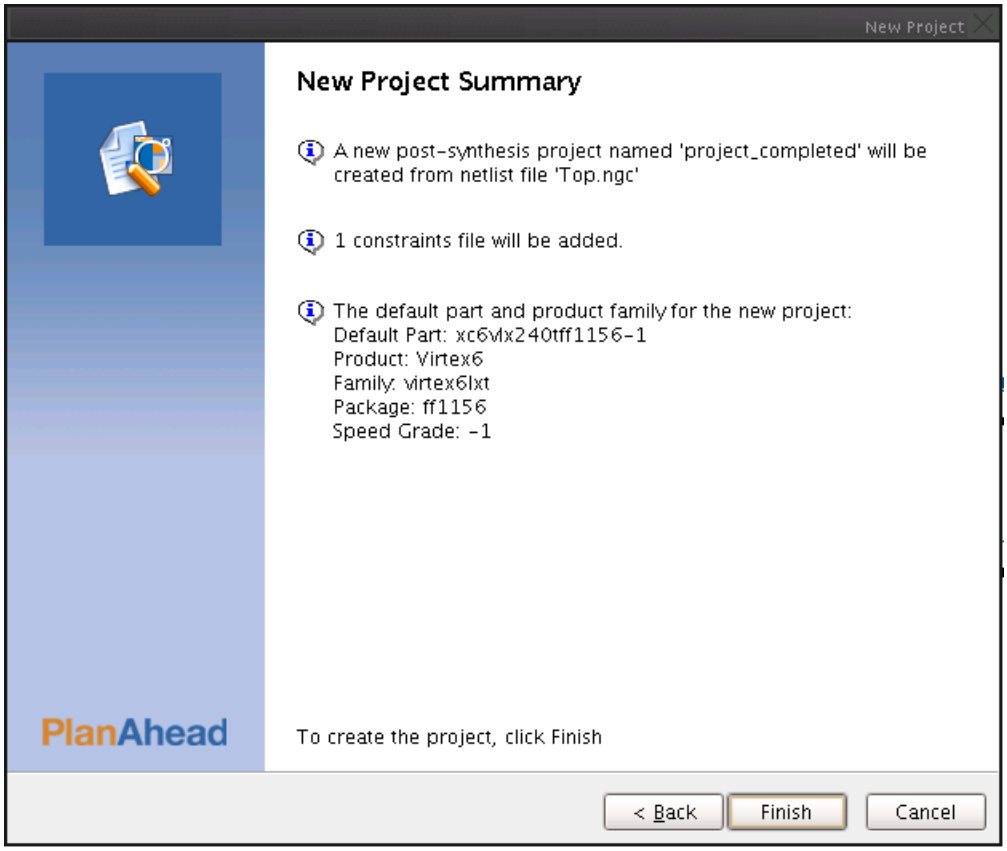

Figure 4: New Project Summary

Step 3: Create RPs and add RM Modules Step 3

3-1. Create the Reconfigurable Partition (RP) for U1_RP_Bram.

3-1-1. Load the Netlist into memory by selecting the Netlist Design from the Flow Navigator.

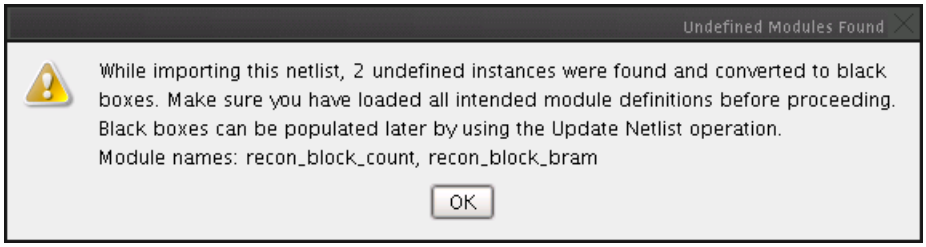

Figure 5: Undefined Modules Warning

The message about undefined instances that appears above is expected because no netlists have been assigned to the Reconfigurable Modules (RMs) yet. Click **OK**.

NOTE: Many of the windows and tools described in this tutorial are only available when the Netlist Design is open. If you close the Netlist Design, or if the project is closed and reopened, you must re-open the Netlist Design by clicking on Netlist Design, as shown above.

3-1-2. In the Netlist window, select and right-click on U1_RP_Bram, and select **Set Partition.** This launches the Set Partition wizard.

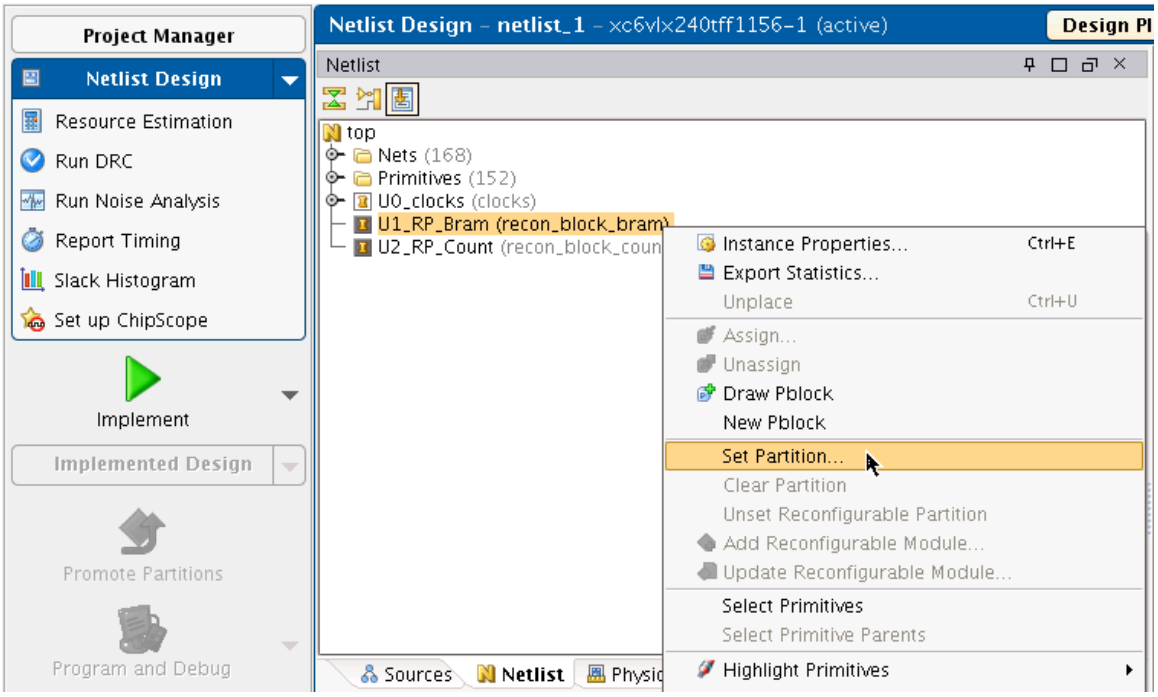

Figure 6: Setting Partition on Reconfigurable Module

3-1-3. Click **Next** on the first screen of the wizard, and select **is a reconfigurable Partition** if not already selected. Click **Next**.

3-1-4. Name the Reconfigurable Module **`BramFirst`**, select **Netlist already available** if not already selected, and click **Next**.

3-1-5. Fill in the Top Netlist File by browsing to **`xpr_bram_led/Implementation/BramFirst/recon_block_bram.ngc`**.

3-1-6. Click **Open**, and then click **Next**.

Note that optional netlist directories could be added here if the RM had lower-level netlists associated with it. In this case, there are none.

3-1-7. Click **Next** again to bypass the optional constraint files screen. This is where module level constraint files could be added. In this case, there are none.

3-1-8. Verify the Set Partition Summary and click **Finish** to complete the wizard.

At this point a Reconfigurable Partition has been created for U1_RP_Bram. The icon in Netlist window has changed from  to  to signify this, and there is one Reconfiguration Module shown under U1_RP_Bram. Also this instance is now shown as a Pblock in the Physical Constraints window.

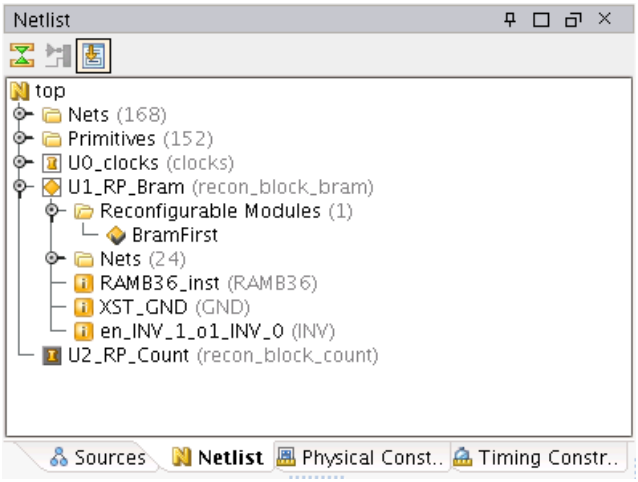

Figure 7: Icon for RP

3-2. Follow the procedure shown in step 3-1 to create an RP for U2_RP_Count.

3-2-1. Select and right-click on U2_RP_Count, and select **Set Partition**.

3-2-2. Complete the Set Partition wizard by setting the following:

- Select **is a reconfigurable partition** if not already set.
- Name the reconfigurable module `CountCW`.
- Select **Netlist already available for this Reconfigurable Module** if not already set.
- Set Top Netlist File to **`xpr_bram_led/implementation/CountCW/recon_block_count.ngc`**.

There is now one RM for each RP in this design. The next step will describe how to add additional RMs for an RP.

Step 4: Add additional RM Modules Step 4

4-1. Add an RM to RP U1_RP_Bram.

4-1-1. In the Netlist window, select and right-click on U1_RP_Bram, and select **Add Reconfigurable Module**. This will launch the Add Reconfigurable Module wizard.

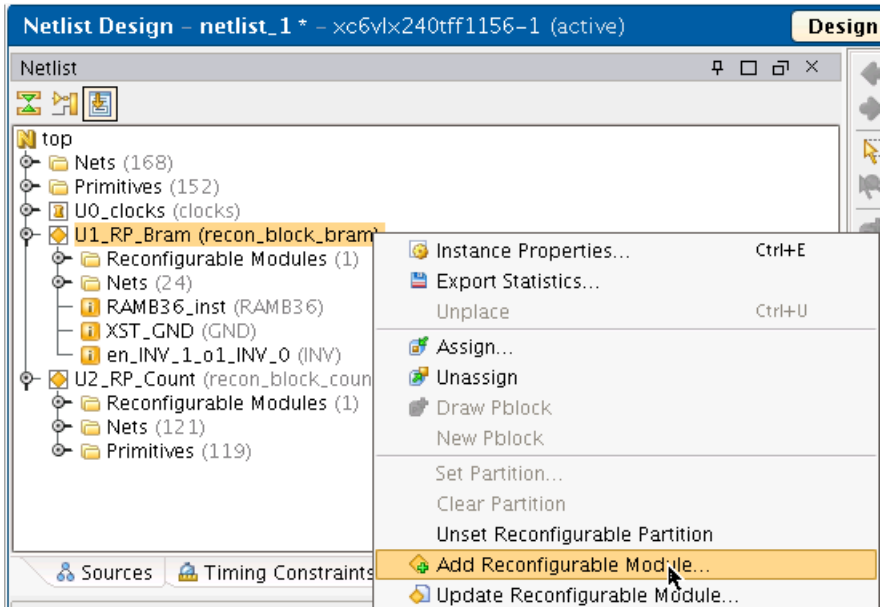

Figure 8: Add Reconfigurable Module

4-1-2. Click **Next** on the introduction page of the wizard.

4-1-3. Name the new Reconfigurable Module **`BramSecond`**, select **Netlist already available for this Reconfigurable Module** if not already selected, and click **Next**.

4-1-4. Set the Top Netlist File by browsing to **`xpr_bram_led/Implementation/BramSecond/recon_block_bram.ngc`** and clicking **Open**.

4-1-5. Again, there are no lower-level netlists to add to the option Netlist directories, so click **Next** to continue.

4-1-6. Click **Next** to bypass the optional module level constraints file screen.

4-1-7. Verify the Add Reconfigurable Module Summary page and click **Finish** to complete the wizard.

4-2. Follow the procedure shown in step 4-1 to add an RM to U2_RP_Count.

4-2-1. Right click on `U2_RP_Count` and select **Add Reconfigurable Module.**

4-2-2. Complete the Add Reconfigurable Module wizard by setting the following:

- Name new Reconfigurable Module **`CountCCW`**.
- Select **Netlist already available for this Reconfigurable Module** if not already selected.
- Set Top Netlist File to `xpr_bram_led/Implementation/CountCCW/recon_block_count.ngc`.

At this point one additional Reconfigurable Module (RM) has been added to each Reconfigurable Partition (RP), and the netlist window should look similar to Figure 9 below. You will notice that under each RP there are Nets and Primitives listed. These nets and primitives are specific to the RM that is currently active, and are signified by the yellow diamond with a checkmark . In Figure 9 the active modules are BramSecond and CountCCW. You can change which RM is active by right-clicking on it and selecting **Set as Active Reconfigurable Module**.

Take a moment to explore the primitives associated with the various RMs. You will notice a RAMB36 for the RMs associated with U1_RP_Bram, and under the Primitives folder of CountCW/CountCCW you will see Slice logic (LUT, XORY, and FDR) and most importantly OBUF primitives. The OBUF primitives are especially important because they will need to be included in the RP Area Group Range in *Step 6 - Floorplanning Reconfigurable Partitions*.

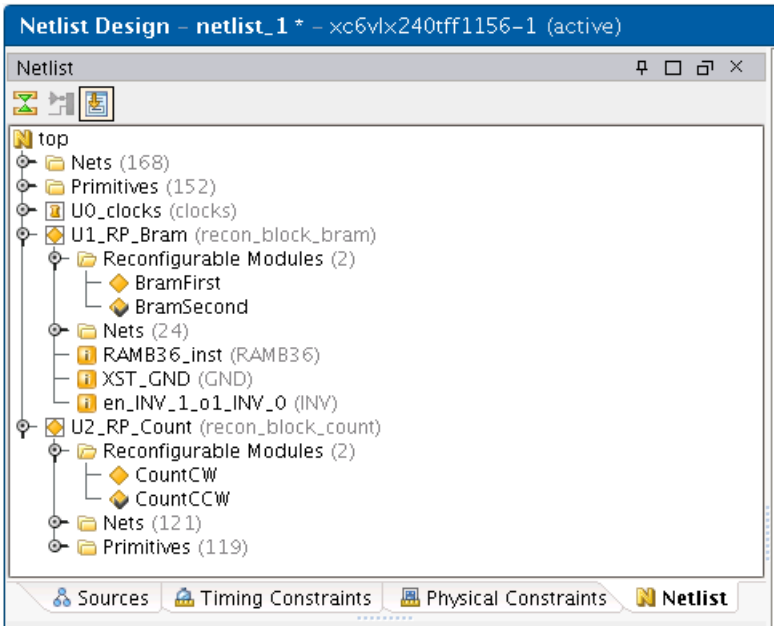

Figure 9: Netlist Window After Adding Reconfigurable Modules

Step 5: Add Black Box Modules (optional) Step 5

This step is optional as black box modules are not required for all PR designs. The purpose of creating a black box module is to generate a "blanking" bit file in the BitGen step. For more information on the uses for this bit file, refer to the *Partial Reconfiguration User Guide* (UG702) available at http://www.xilinx.com/tools/partial-reconfiguration.

5-1. Add a Black Box (BB) module to U1_RP_Bram.

5-1-1. In the Netlist window, right-click on U1_RP_Bram, and select **Add Reconfigurable Module**. This launches the Add Reconfigurable Module wizard.

5-1-2. Click **Next** on the introduction page of the wizard.

5-1-3. Name the new Reconfigurable Module **`BramBB`**, select **Add this Reconfigurable module as a black box without a netlist**, and click **Next**.

5-1-4. Because there will be no netlist or constraint file associated with a black box module, the wizard does not prompt you for any of this information.

Verify the Add Reconfigurable Module Summary page and click **Finish** to complete the wizard.

5-2. Follow the procedure shown in step 5-1 to add a BB module to U2_RP_Count.

5-2-1. In the Netlist window, right-click U1_RP_Bram and select **Add Reconfigurable Module.** This will launch the Add Reconfigurable Module wizard.

5-2-2. Complete the Add Reconfigurable Module wizard by setting the following:

5-2-3. Set Reconfigurable Module Name to **`CountBB`**.

5-2-4. Select **Add this Reconfigurable module as a black box without a netlist**.

If you completed this optional step, there will now be one black box module under each Reconfigurable Partition (RP) in the netlist window as shown in Figure 10. Here you will notice that there are no longer any Nets or Primitives listed for the RM. This is because the black box RM is currently active, and there is no logic or nets associated with this module.

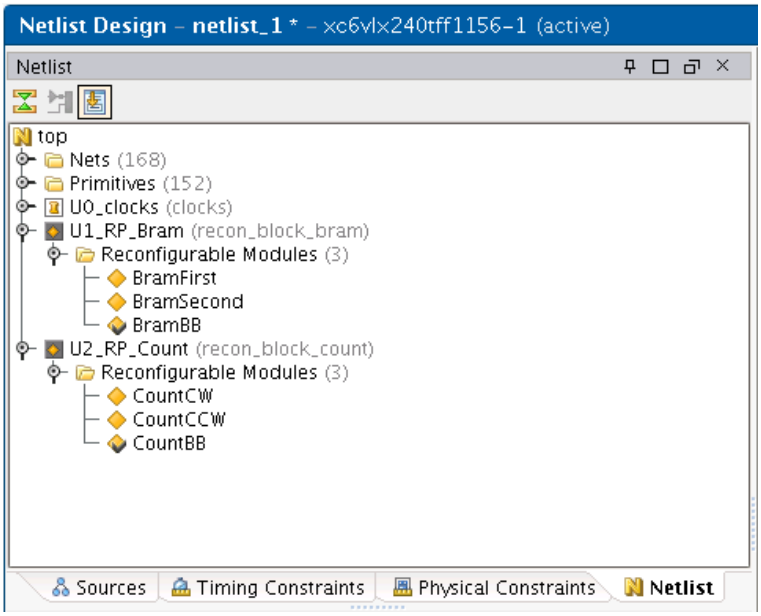

Figure 10: Netlist Window with Black Box Modules

Step 6: Floorplanning Reconfigurable Partitions Step 6

Each Reconfigurable Partition (RP), `U1_RP_Bram` and `U2_RP_Count`, in the example design must have AREA_GROUP (AG) Range constraints to designate which physical resources are part of that RP.

All physical resources not part of the AG Range constraint associated with an RP are part of the static logic. (Static logic is unaffected by partial reconfiguration and remains operational during the reconfiguration process.) The AG Range constraints should not be created until the RP has been created with the Set Partition command as described in *Step 3 - Create RPs and Add RM Modules*.

6-1. Create the AG Range for pblock_U1_RP_Bram.

6-1-1. Make BramFirst and CountCW the active Reconfigurable Modules (RMs) by right-clicking them in the Netlist window and selecting **Set as Active Reconfigurable Module**.

Since the resources required for a particular RM depend on which RM is active, it is important to make sure that a black box RM is not active or PlanAhead will not be able to report the correct resources required for the AG Range. For situations where different RMs associated with an RP use various resources, an RP's AG Range must contain a superset of resources used by all its RMs.

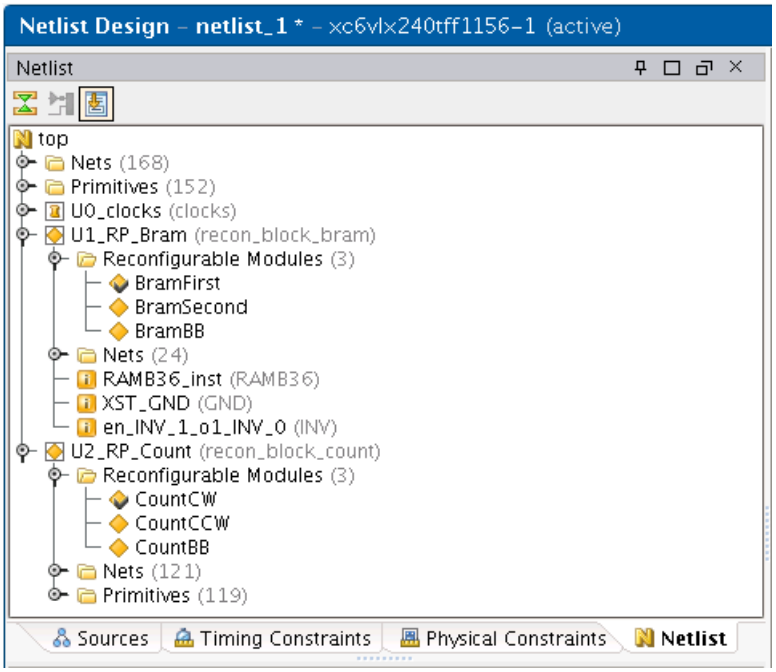

Figure 11: Setting Active Reconfiguration Modules

6-1-2. Now click on the Physical Constraints window to get a list of all the current Pblocks. PlanAhead refers to AREA_GROUP constraints as Pblocks, and automatically creates Pblocks for any modules defined as an RP. In the Physical Constraints window, select the Reconfigurable Partition, **`pblock_U1_RP_Bram`**.

6-1-3. On the left hand side of the Device view, click the Set Pblock Size button (

). This can also be done by right-clicking **`pblock_U1_RP_Bram`** and selecting **Set Pblock Size**.

6-1-4. Draw a box that encompasses some slice logic and at least one RAMB36 (pink columns).

6-1-5. After drawing the rectangle, select both **SLICE** and **RAMB36** resources for the AG, as shown in Figure 12.

6-1-6. Click **OK**.

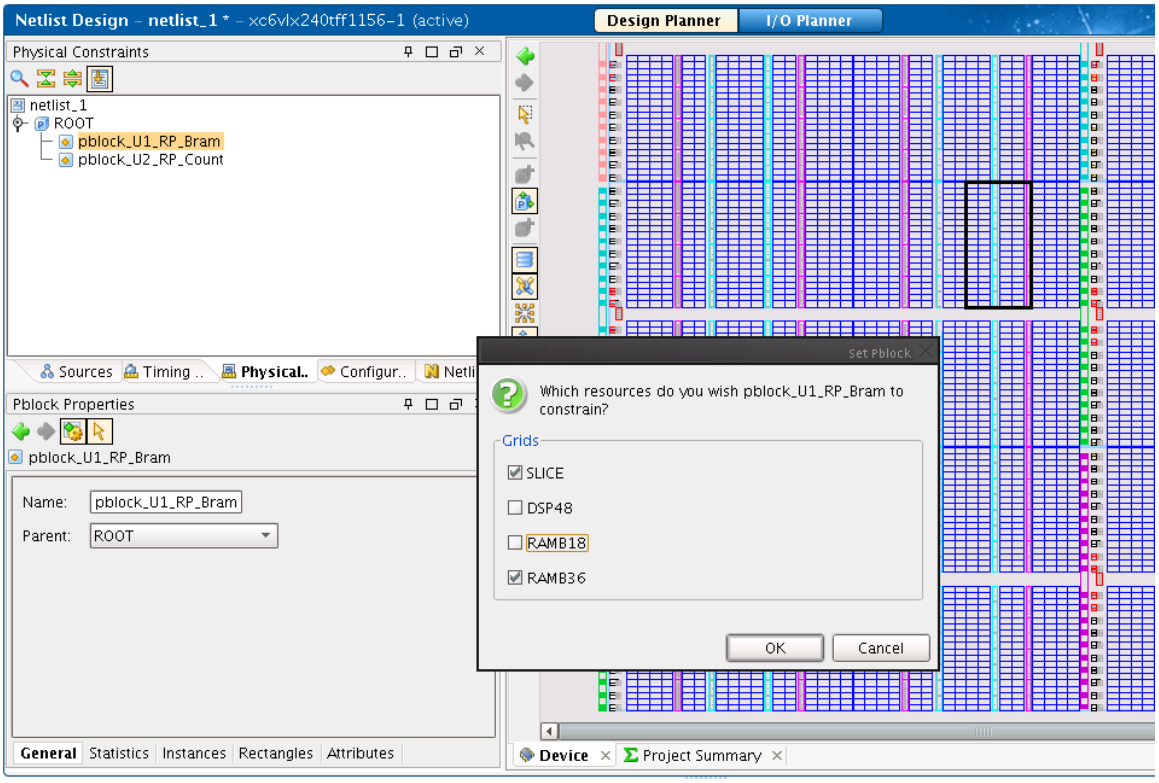

Figure 12: Setting Pblock Resources (pblock_U1_RP_Bram)

6-2. Follow the procedure shown in step 6-1 to create an AG Range for pblock_U2_RP_Count.

6-2-1. With pblock_U2_RP_Count selected, use the Set Pblock Size tool and draw a box that encompasses slice logic as well as IO logic. Because U2_RP_Count already has I/O pin placement in the UCF, the AG Range must include pins AD21, AH27, AE21 and AH28, which are visible in the floorplan.

6-2-2. After drawing the rectangle, select **SLICE** and **IOB** resources for the AG as shown in Figure 13.

6-2-3. Click **OK**.

If the RM had input or output flip-flops (or other input/output logic) then other resources such as ILOGIC and OLOGIC would be included as well. This design does not contain such resources.

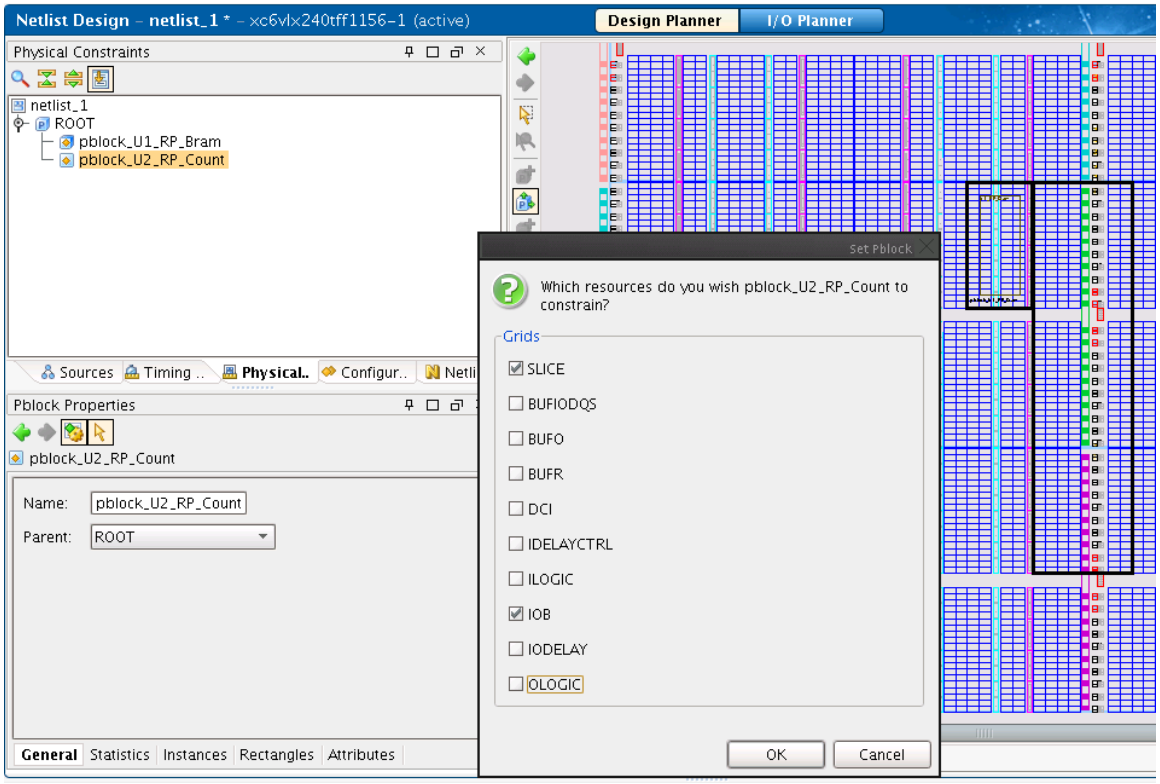

Figure 13: Setting Pblock Resources (pblock_U2_RP_Count)

Review floorplan and make any necessary adjustments.

6-2-4. The resulting floorplan should look similar to the image in Figure 14.

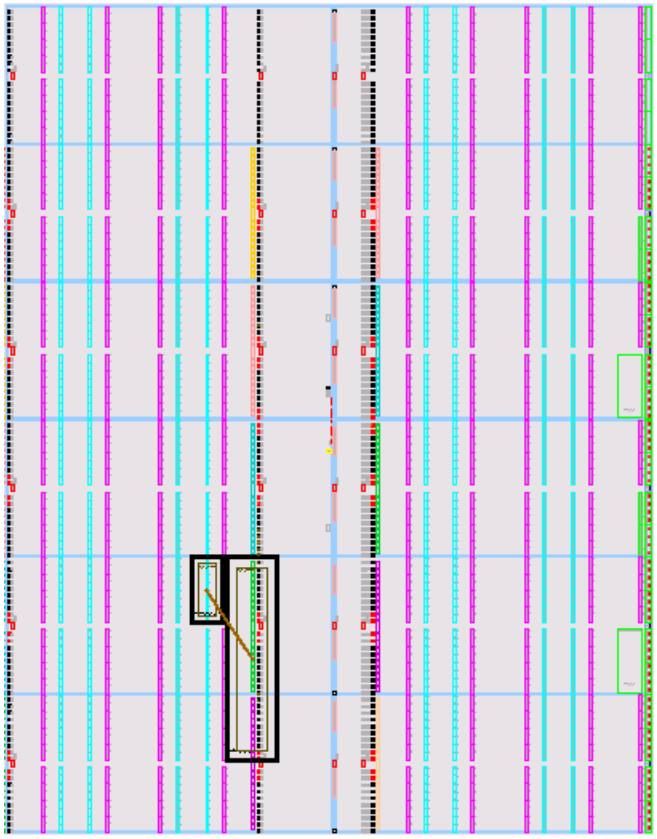

Figure 14: Final AG Ranges

6-2-5. Verify the pins required for U2_RP_Count are within the AG Range. The pins AD21, AH27, AE21 and AH28 must be included in the rectangle of the AG Range for U2_RP_Count as shown in Figure 14. Also note that the rectangle extends to the right of the IO column. This is critical as there are I/O routing resources on the right side of this IO column that must be included in the AG Range or the router will fail implementation. This is because all RM routing resources must be within the RP region as defined by the AG Range.

To locate these pins in the Device view in order to verify they are inside of the AG Range, select **Edit > Find** and search for **Site with Name matches AD21**. The pin will be highlighted in the Device view since the UCF contains a LOC constraint already. Select **View > Fit Selection** (F9) to zoom to the highlighted site. The same procedure can be used to find the pins AH27, AE21, and AH28.

6-2-6. Verify the RAMB36 required for U1_RP_Bram is included in the AG Range. Select the pblock_U1_RP_Bram from the Physical Constraints window, and view the Statistics shown in the Pblock Properties window, as shown in Figure 13. Note that the available number of RAMBFIFO36E1 exceeds the required amount.

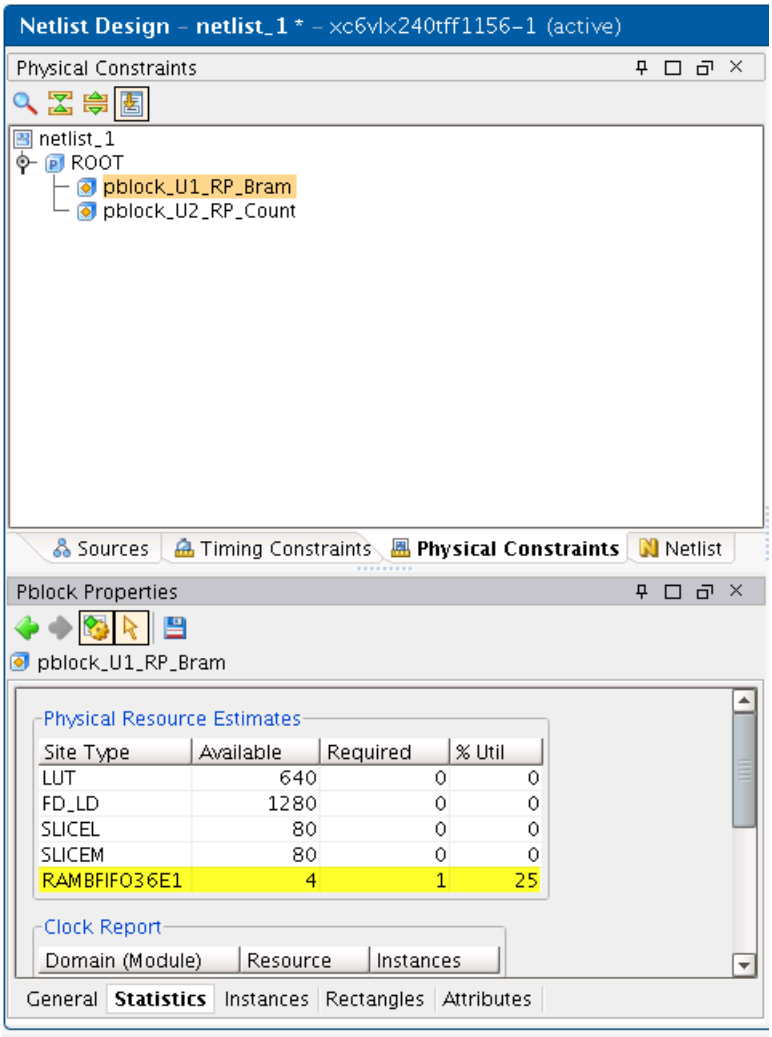

Figure 15: Pblock Statistics (pblock_U1_RP_Bram)

6-2-7. The Area Group Range constraints created by previous steps can be viewed in PlanAhead. First save the Design by selecting **File > Save Design**.

6-2-8. Click on the Project Manager in the Flow Navigator (left hand side of PlanAhead).
This will bring up the Sources window where all the Design Sources and Constraints can be seen.

6-2-9. Under Constraints, find the UCF and double click it to open it.
You should see Area Group constraints similar to the following.

```
INST "U1_RP_Bram" AREA_GROUP = "pblock_U1_RP_Bram";
AREA_GROUP "pblock_U1_RP_Bram" RANGE=SLICE_X48Y60:SLICE_X55Y79;
AREA_GROUP "pblock_U1_RP_Bram" RANGE=RAMB36_X3Y12:RAMB36_X3Y15;
INST "U2_RP_Count" AREA_GROUP = "pblock_U2_RP_Count";
AREA_GROUP "pblock_U2_RP_Count" RANGE=SLICE_X56Y20:SLICE_X67Y79;
AREA_GROUP "pblock_U2_RP_Count" RANGE=IOB_X1Y20:IOB_X1Y79;
```

Step 7: Partition Pins and RP Interface Timing Step 7

Partition Pins (PP) are required in the Partial Reconfiguration flow for all Reconfigurable Partition (RP) interface signals that are not global logic or dedicated routes. Partitions provide a known routing connection to the RP, and are automatically inserted by NGDBuild when implementation is run (these are a replacement technology for bus macros from the previous PR flow).

The current implmentation of PP requires proxy logic, which is a LUT1. The LUT1 is being inserted in the input and output paths of the RP, and it is recommended to register these inputs/outputs on both sides of the RP boundary. This will help minimize any timing closure issues related to the RP interface. If these guidelines are followed it is likely that a simple period constraint will be enough to constrain this interface. However, in situations with very tight timing requirements it may be necessary to create TPSYNC constraints on the PP, or add LOC constraints to the static logic to minimize routing delays between the static logic and the PP. For more information on adding TPSYNC constraints to the partition pins, see the *Partial Reconfiguration User Guide* (UG702) available at http://www.xilinx.com/tools/partial-reconfiguration.

Because the same static logic implementation will be used for every configuration, it is important to try and meet timing with the most timing critical Reconfigurable Modules (RMs) first. The RP interfaces in this tutorial are very similar between the various RMs, and a global PERIOD constraint is enough to meet the RP interface timing.

7-1. Add a PERIOD to constrain the design (including the RP interfaces).

7-1-1. From the Netlist Design window, click the Timing Constraints tab to open this window.

7-1-2. Open the New Timing Constraint tool by right-clicking in the window and selecting **New Timing Constraint,** or by clicking the New Timing Constraint button  at the top of the window.

7-1-3. Under the Basic group (TNM) category, set the following values:

- Group name: Enter **`clk_p`**
- Group type: Set to **Net**
- TNM type: Set to **TNM_NET**
- Predefined group: Leave blank
- Net: Set to **clk_p**

7-1-4. Click **OK** to add the constraint.

7-1-5. Open the New Timing Constraint tool again. Under the Timespec period category, set the following values:

- TimeSpec name: Enter **`TS_clk_p`**
- Period: Set to **5 ns**
- Click the browse button and set **User defined** in the Group constraints type field, and **clk_p** in the User defined groups field.

7-1-6. Click **OK** to add the constraint. The Timing Constraint window should now look like Figure 16. Note that this is a global timing constraint that will constrain all synchronous paths connected to clk_p in the design. This is not a timing constraint specific to the RP interface.

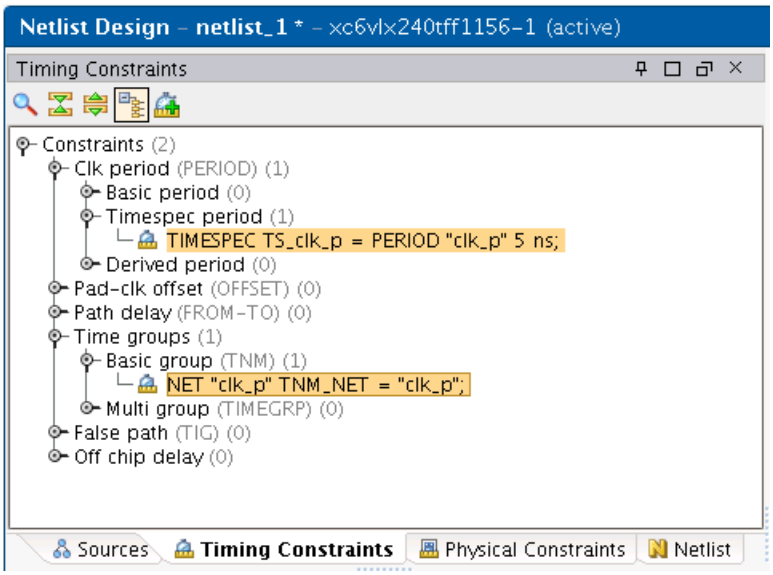

Figure 16: Timing Constraints View (pblock_U1_RP_Bram)

7-1-7. Save the Design and navigate back to the Project Manager to view the UCF. The following constraints should now be included in the constraint file.

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

Step 8: Partial Reconfiguration Design Rule Checks Step 8

There are many Partial Reconfiguration specific design rules that must be followed in order to implement a valid design. Some of these rules have been incorporated in the PlanAhead DRC engine under the Partial Reconfig and Partition headings. These checks should be run on the PR design before implementing configurations and generating bit files.

On a normal design you might want to run all the PlanAhead DRCs. For this tutorial, we will run the Partial Reconfig and Partition DRCs only.

8-1. Run the Partial Reconfiguration and Partition DRCs.

8-1-1. From the Flow Navigator, locate the Run DRC button under the Netlist Design (Figure 15).

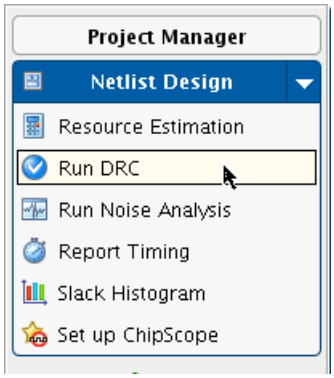

Figure 17: Run DRC Button

8-1-2. From the Run DRC dialog box, select the **Partition** and **Partial Reconfig** rules.

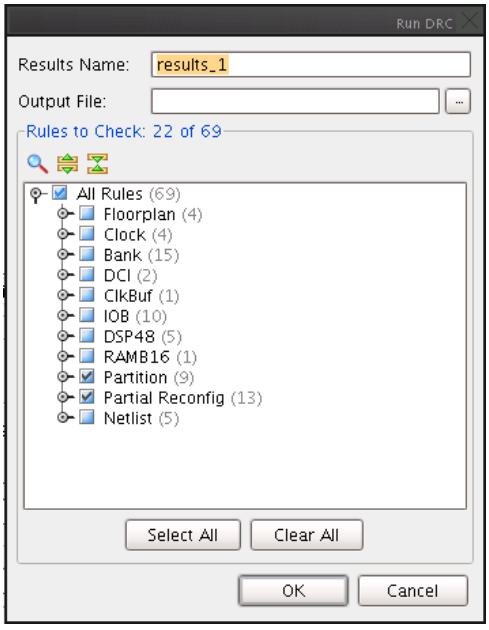

Figure 18: Partition and Partial Reconfig DRCs

8-1-3. Review the messages returned by the DRC in the Details column, and note the Severity. The messages returned by the DRC can be Advisory, Warning, Error, or Fatal. In this case, the worst Severity returned is a Warning (Figure 19).

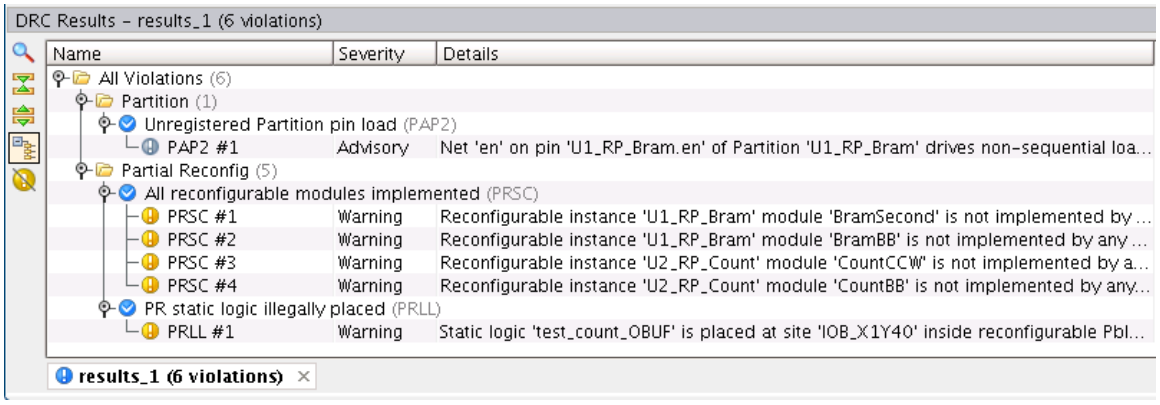

Figure 19: DRC Results

Step 9: Implement and Promote a Configuration Step 9

Each Reconfigurable Partition (RP) may have multiple Reconfigurable Modules (RMs) associated with it, yet only one RM can be implemented at any one time for the RP.

The set of active RMs along with static logic is called a configuration and is a complete design. Multiple configurations will exist for a Partial Reconfiguration project so that different permutations of RMs can be implemented, thus generating full and partial bit files. Each configuration is its own independent implementation run with resulting output files, such as `.ngd`, `.ngm`, `.ncd`, `.pcf` and report files. Xilinx software tools and debug techniques can be applied to each configuration individually, such as opening a specific configuration's `.ncd` with FPGA Editor or doing a gate-level simulation.

The design in this tutorial could be fully implemented with only two configurations using these RM 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
```

The full bit file `config_1.bit` contains RMs BramFirst and CountCW while the full bit file `config_2.bit` contains BramSecond and CountCCW.

Two other configuration sets are possible that will generate unique full bit files. However, since they reuse modules previously implemented, the partial bit files will be 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
```

The implementation of a configuration within the PlanAhead software is called a "run". A "run" must be created for each configuration.

PlanAhead automatically creates a configuration when the project is created. The RMs set for this configuration depend on the order the RMs were added to the project as the first RM defined for each RP gets set for this configuration (BramFirst and CountCW in the case of this tutorial).

An FPGA configured with a full bit file contains the RMs that were implemented in the configuration. If the system requires that only the static logic is functioning after loading the full bit file, then implement a configuration containing black boxes for all Reconfigurable Partitions. The resulting partial bit files are effectively blank bit files for the Reconfigurable Partitions.

9-1. Implement configuration config_1.

9-1-1. Verify the RMs set for `config_1` are BramFirst and CountCW. In the Design Runs window (**Window > Design Runs**), select **config_1**. In the Implementation Run Properties window, select the Partitions tab (Figure 18). If the Module Variants listed are not BramFirst and CountCW, change them so they are.

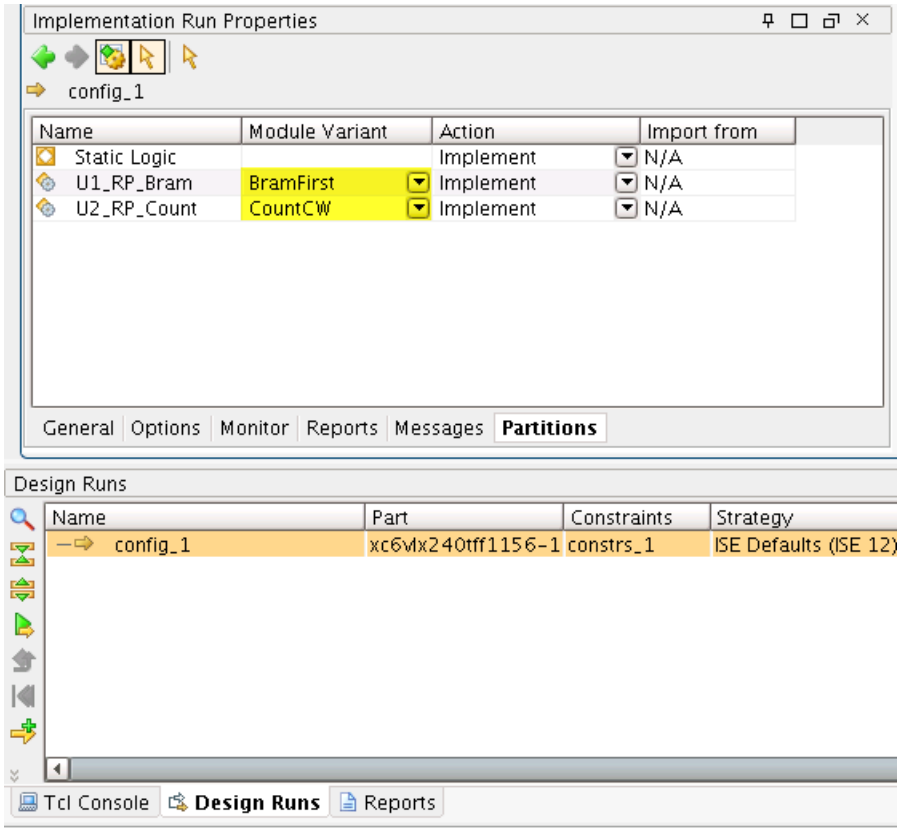

Figure 20: Configuration Module Variants (config_1)

9-1-2. Click the **Implement** button in the Flow Navigator to launch the implementation run.

The Design Runs window Status field shows when NGDBuild, Map, PAR, and TRCE are running. This can also be monitored from the status bar in the upper right-hand corner of PlanAhead, or for more details utilize the Compilation Log window.

9-1-3. Promote configuration “config_1”.

Now that “config_1” has been successfully implemented, it can be promoted. Note that other configurations can be run without promoting the first configuration, but this will result in incompatible partial bit files between configurations. One configuration must be chosen to be promoted, and all other configuration must “import” the static logic from this promoted partition in order to get compatible partial bit files between multiple configurations. This will guarantee consistent proxy logic between all configurations.

The compatibility between multiple configurations can be checked using PR Verify, which is shown in Step 11 – Run PR Verify.

9-1-4. From the Flow Navigator, click the Promote Partitions button (Figure 21).

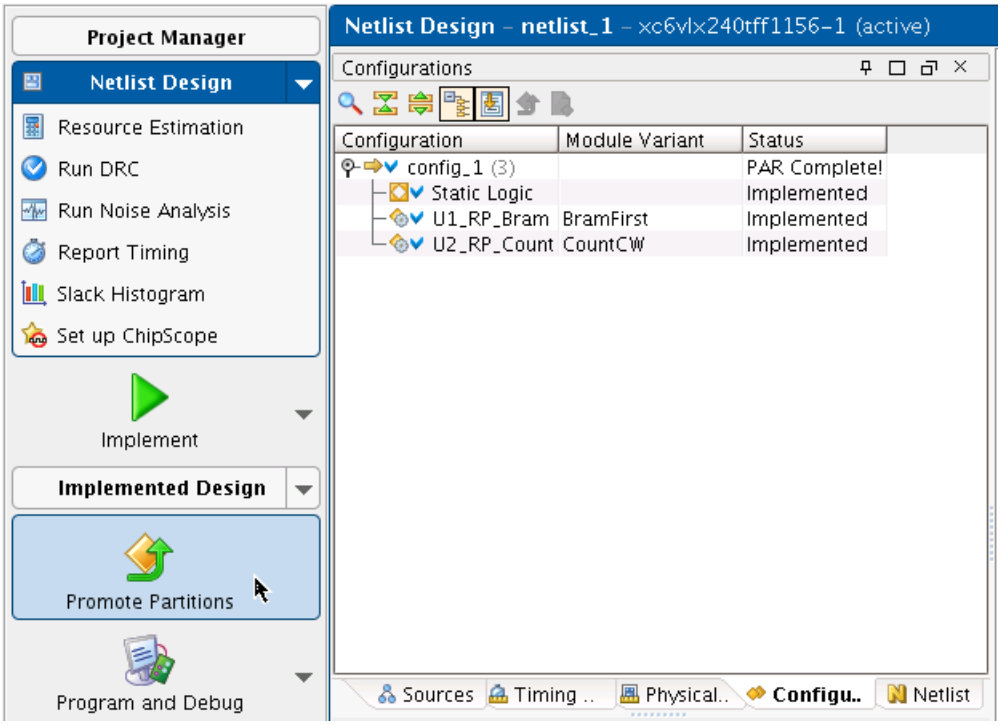

Figure 21: Promote Partitions Button

9-1-5. In the Promote Partitions dialog box, click **OK**.

9-1-6. From the Netlist Design view, open the Configurations tab and view and note the Status has changed from “Implemented” to “Promoted” (Figure 20).

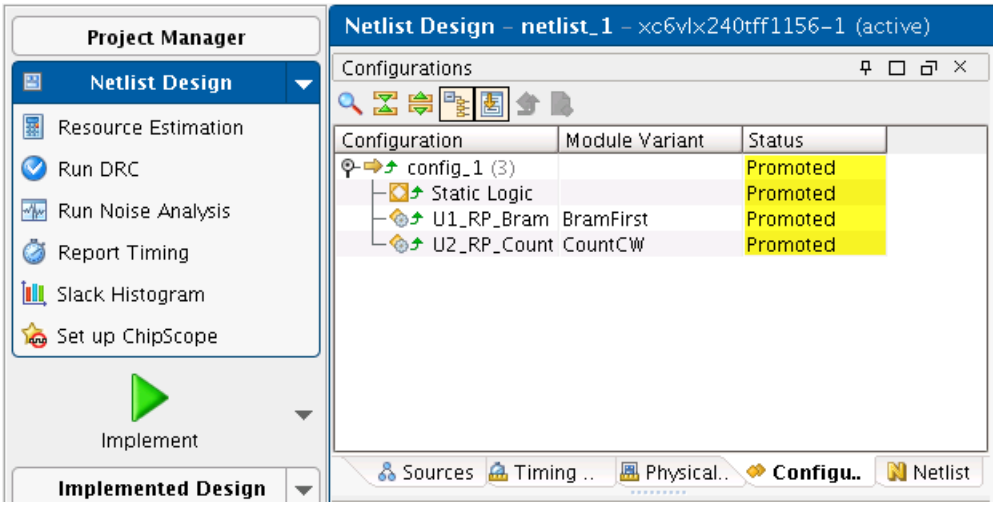

Figure 22: Promoted Status

Step 10: Create and Implement additional Configurations Step 10

10-1. Create a new configuration.

10-1-1. From the Flow Navigator, use the Implement drop-down list to select **Create Multiple Runs.** (Figure 23).

The Create Multiple Runs wizard opens.

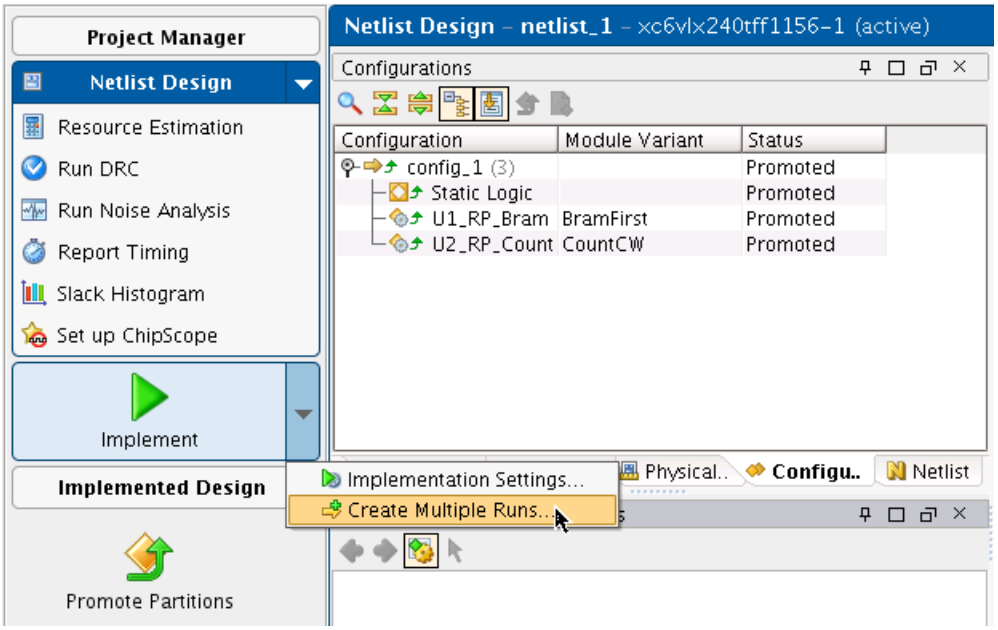

Figure 23: Create Multiple Runs

10-1-2. Create a new configuration.

10-1-3. On the introduction page of the wizard, click **Next**.

10-1-4. On the Setup Implementation Run page, click **Next**.

10-1-5. On the Choose Implementation Strategies and Reconfigurable Modules page, you can create multiple configurations, choose implementation strategies, and define which Reconfigurable Modules (RMs) will make up the configurations. There is already a new configuration listed called “config_2”. This could be renamed, but we’ll leave it as “config_2” for this tutorial.

10-1-6. Click the <Partition_Name> button to open the Specify Partition dialog box (Figure 24).

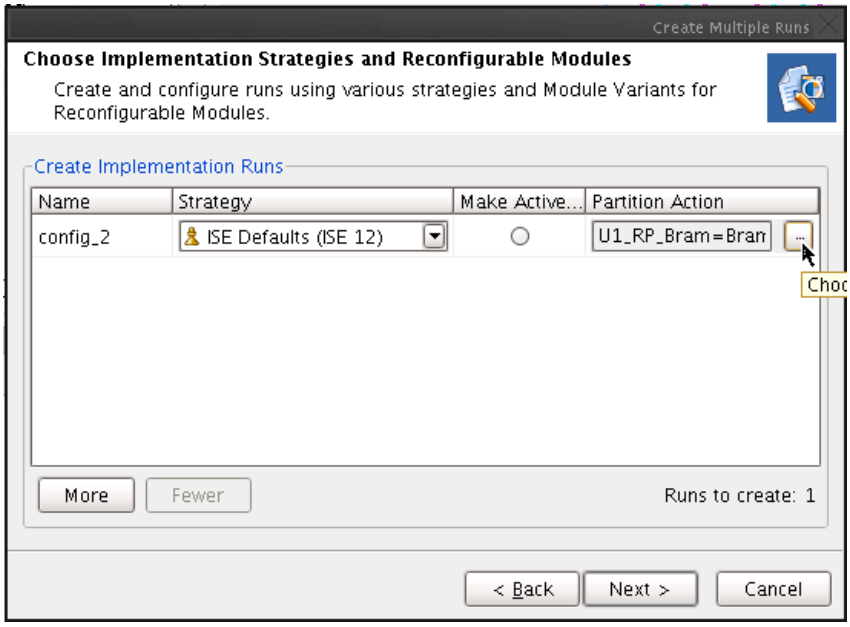

Figure 24: Create Implementation Runs

10-1-7. The default Module Variant is based on which RMs are currently active in the project. In this case it is the BramFirst and CountCW. Because these have already been implemented and imported, they are set to "Import" and have an import location set.

10-1-8. To create a configuration that implements the BramSecond and CountCCW RMs, change the Module Variant column to match these RMs (Figure 25).

Because these RMs have not been implemented (or promoted), the Action field changes to "Implement".

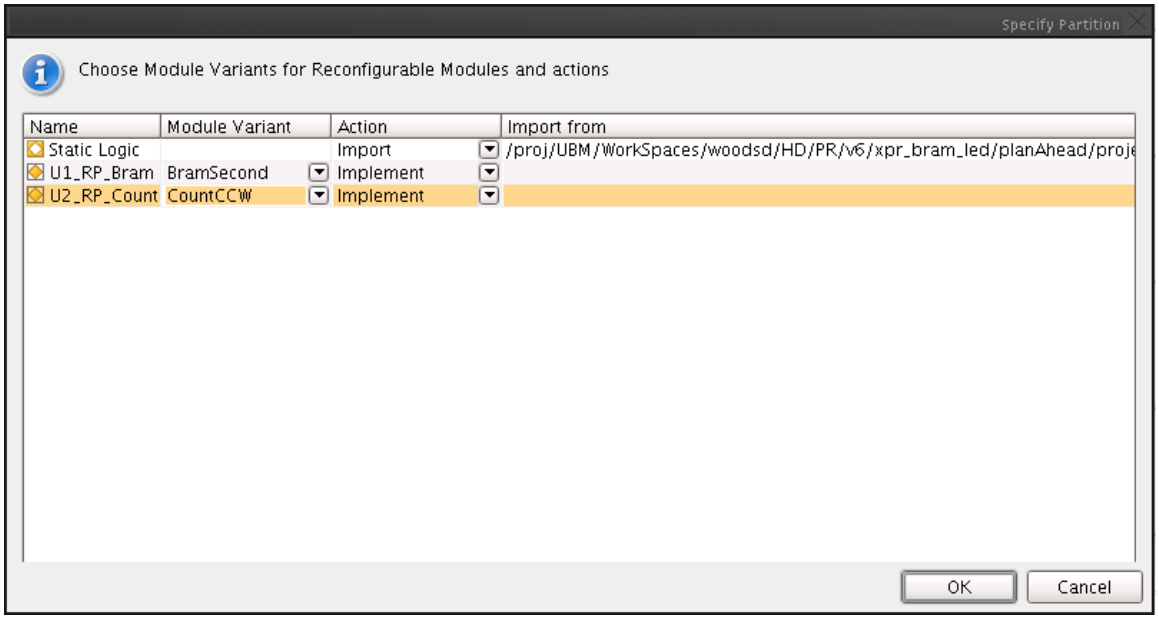

Figure 25: Choose Configurations Dialog Box (BramSecond/CountCCW)

10-1-9. In the Specify Partition dialog box, click **OK**.

10-1-10. Optional. If you did the optional step of creating Black Box module variants, then you can now create an additional configuration to implement those modules. Click **More** in the Choose Implementation Strategies and Reconfigurable Modules page to add **`config_3`**. Click **Choose Configurations** and set the Module Variants to **BramBB** and **CountBB**.

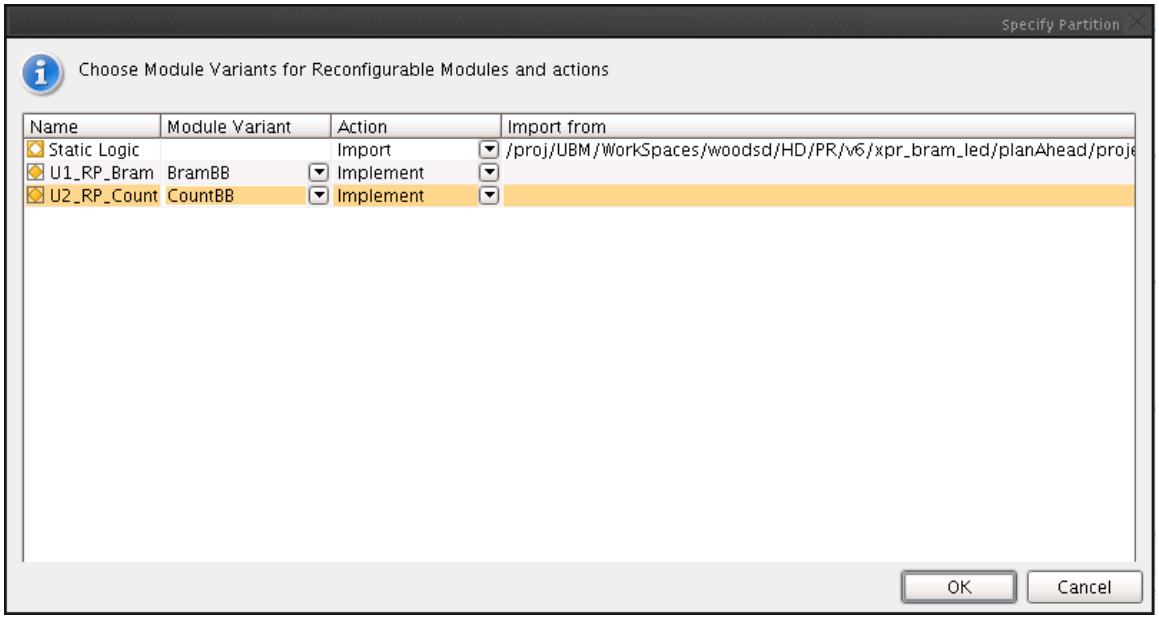

Figure 26: Choose Configurations Dialog Box (BramBB/CountBB)

10-1-11. Click **Next** on the Create Implementation Run page to continue through the wizard.

10-1-12. On the Launch Options page, select the **Do not launch now** radio button and click **Next**. The configurations could have been launched from here, but for the purposes of discussion, we'll launch new configurations in the next couple of steps.

10-1-13. Click **Finish** on the Create Multiple Runs Summary page to complete the wizard.

10-1-14. In the Design Runs window, you will now see new configurations that were created through the wizard. Select a new configuration in the Design Runs window and click the Partitions tab in the Implementation Run Properties window (Figure 27) to verify the Module Variants and Action fields. Note that the Static logic is set to Import, and will be imported from the promoted results from config_1.

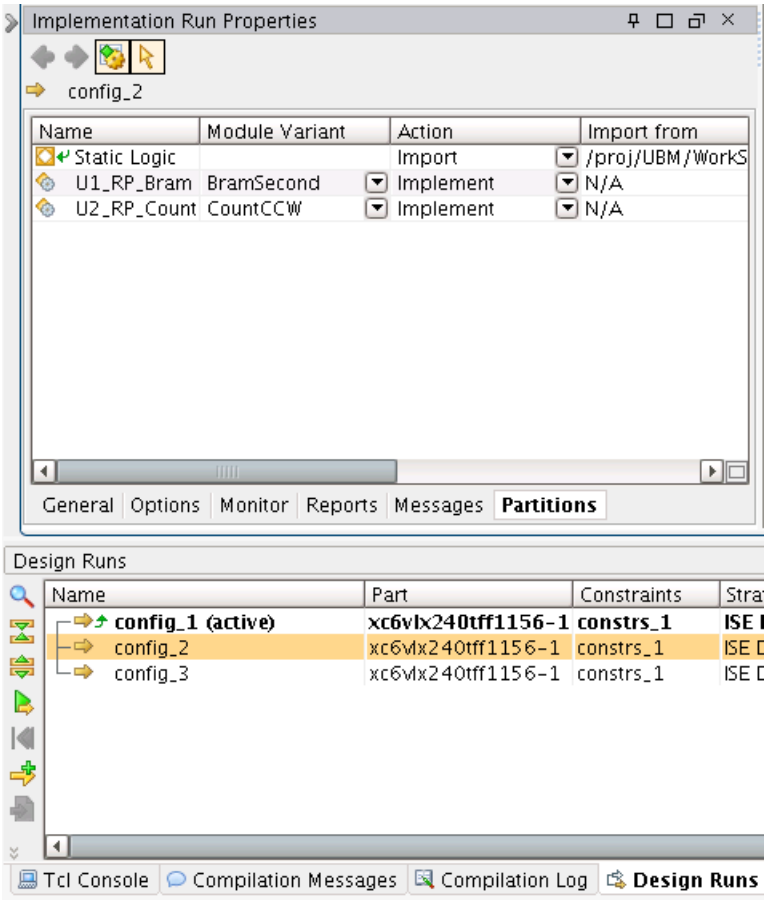

Figure 27: Verify Configuration Settings

10-1-15. Launch a configuration by right-clicking, and choosing **Launch Runs.** Note that if you have multiple new configurations, they can be launched in parallel (on multiple processors, if available) because the results are not dependent on each other like they were for config_1 (Figure 28).

10-1-16. In the Launch Selected Runs dialog box, select the **Launch runs on Local Host** radio button, select the appropriate number of jobs (number of processors to use), and click **OK**.

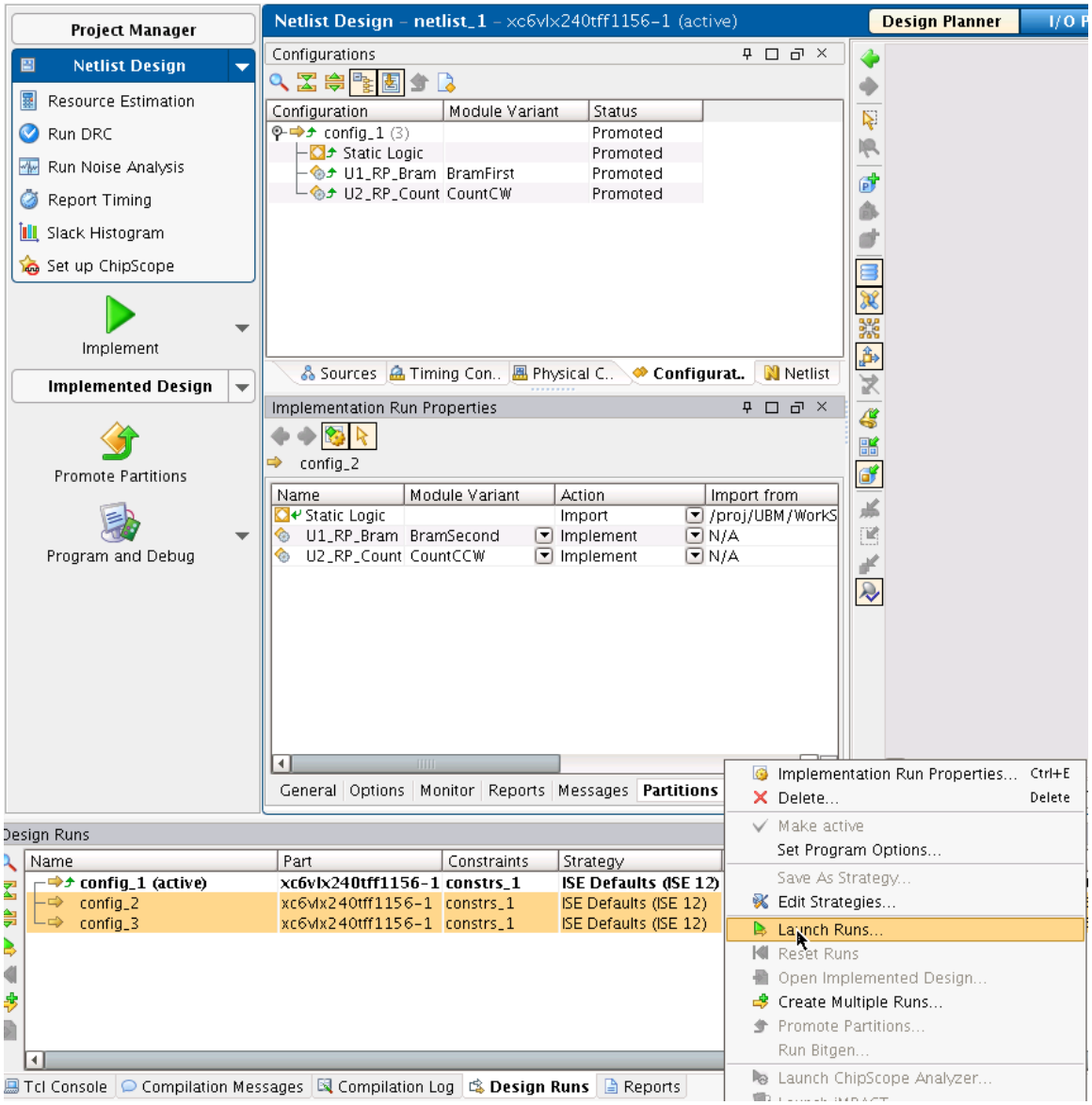

Figure 28: Launch Runs

Step 11: Run PR Verify Step 11

After multiple configurations are implemented, you can compare them to verify that the static logic and partitions pins are consistent across all configurations. This is a check that must be done to ensure the bit files will be compatible.

11-1. Run PR Verify on all configurations.

11-1-1. From the Flow Navigator, use the Program and Debug drop-down box to select **Verify Configuration** (Figure 29).

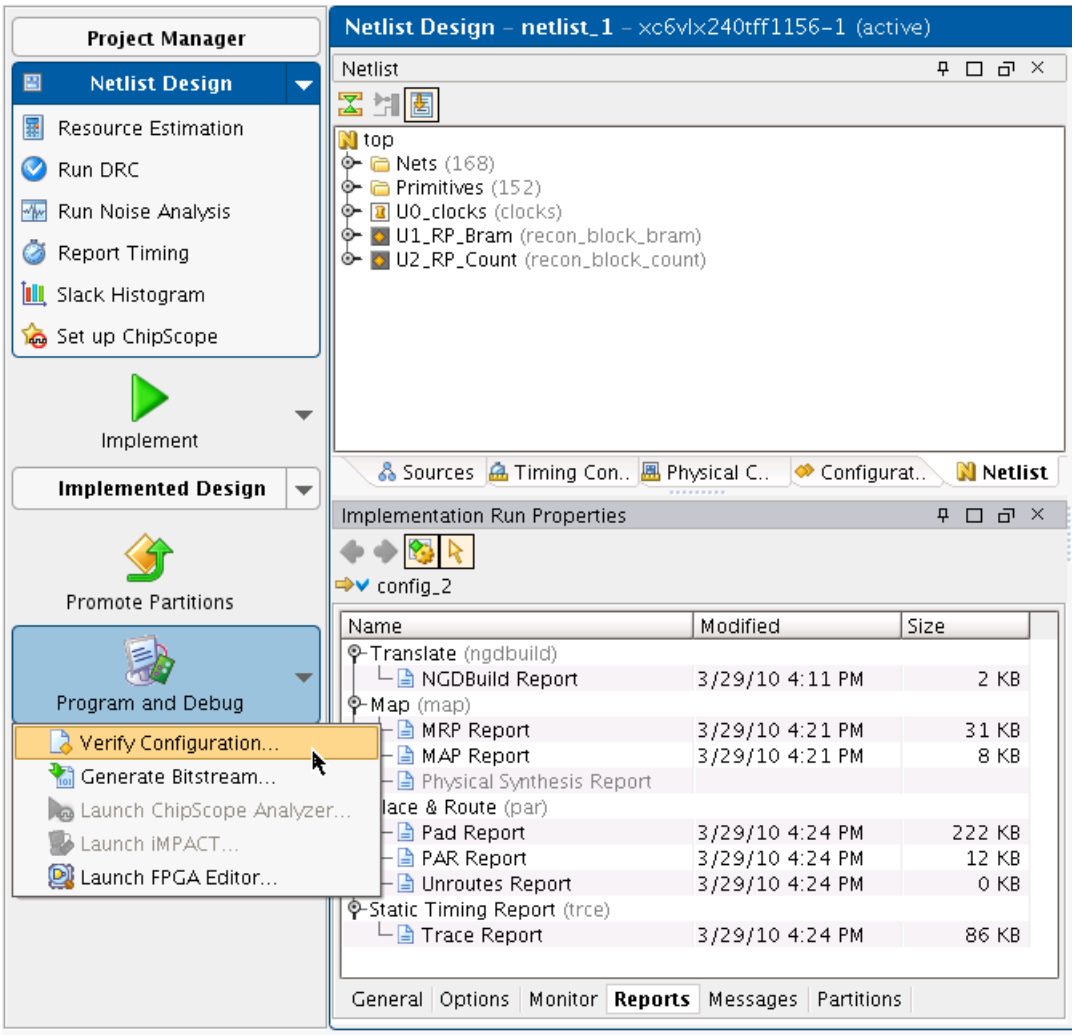

Figure 29: Verify Configuration Settings

11-1-2. Select at least two configurations and up to all configurations to verify against each other, and click **OK** to launch PR Verify.

If the verification checks pass, PlanAhead reports that no errors were found in a message box. This indicates that it is now okay to generate bit files. Click **OK** to close the message box.

The detailed report is opened in PlanAhead and saved to `xpr_bram_led/PlanAhead/project_1/project_1.runs/pr_verify.log`.

Step 12: Generate and Download Bit Files Step 12

For each configuration, multiple bit files will be generated. There will be one full bit file that can be used to program the FPGA during power up, and then one partial bit file for each Reconfigurable Partition (RP) that contains the logic for the module variants associated with the particular configuration. In this tutorial we create configurations with BramFirst, CountCW, BramSecond, CountCCW, and optionally BramBB and CountBB. By generating bit files for each configuration we will end up with partial bit files for all of these Reconfigurable Modules (RMs). Regardless of the full bit file used to configure the device initially, any of the partial bit files can be used to reconfigure the associated PR regions.

12-1. Generate bit files for all the configurations.

12-1-1. In the Design Runs window, highlight all configurations. Right-click in the window and select **Run Bitgen** (Figure 30).

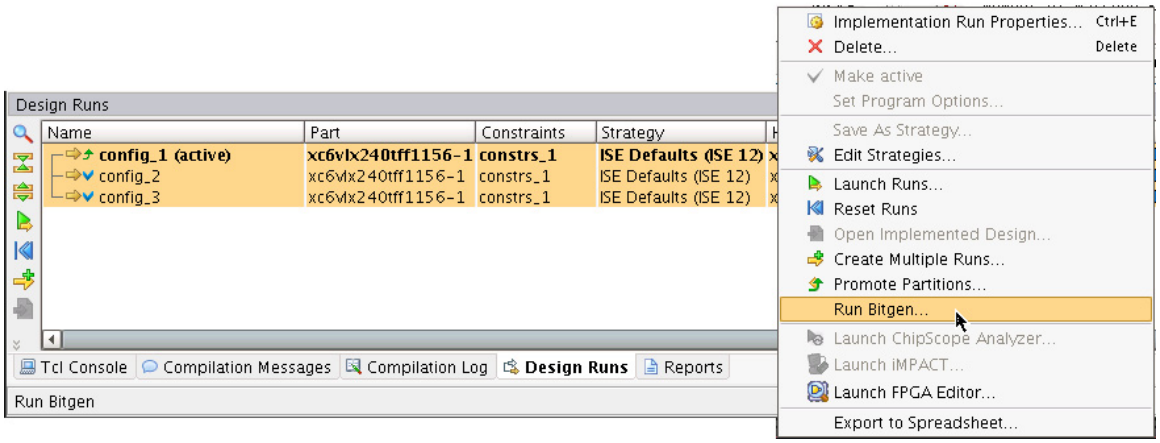

Figure 30: Running BitGen

12-1-2. There are no special BitGen options required to generate partial bit files, so click **OK** on the BitGen options dialog box to launch BitGen.

The partial bit files are downloaded to the FPGA in the same manner as full bit files. The iMPACT software tool can be used for bit file download, verification, and debug purposes in a lab environment.

12-1-3. Connect the USB download cable to the ML605 board and the PC.

12-1-4. Start iMPACT in standalone mode.

12-1-5. From the Flow Navigator, use the Program and Debug drop down box to select **Launch iMPACT** (Figure 31).

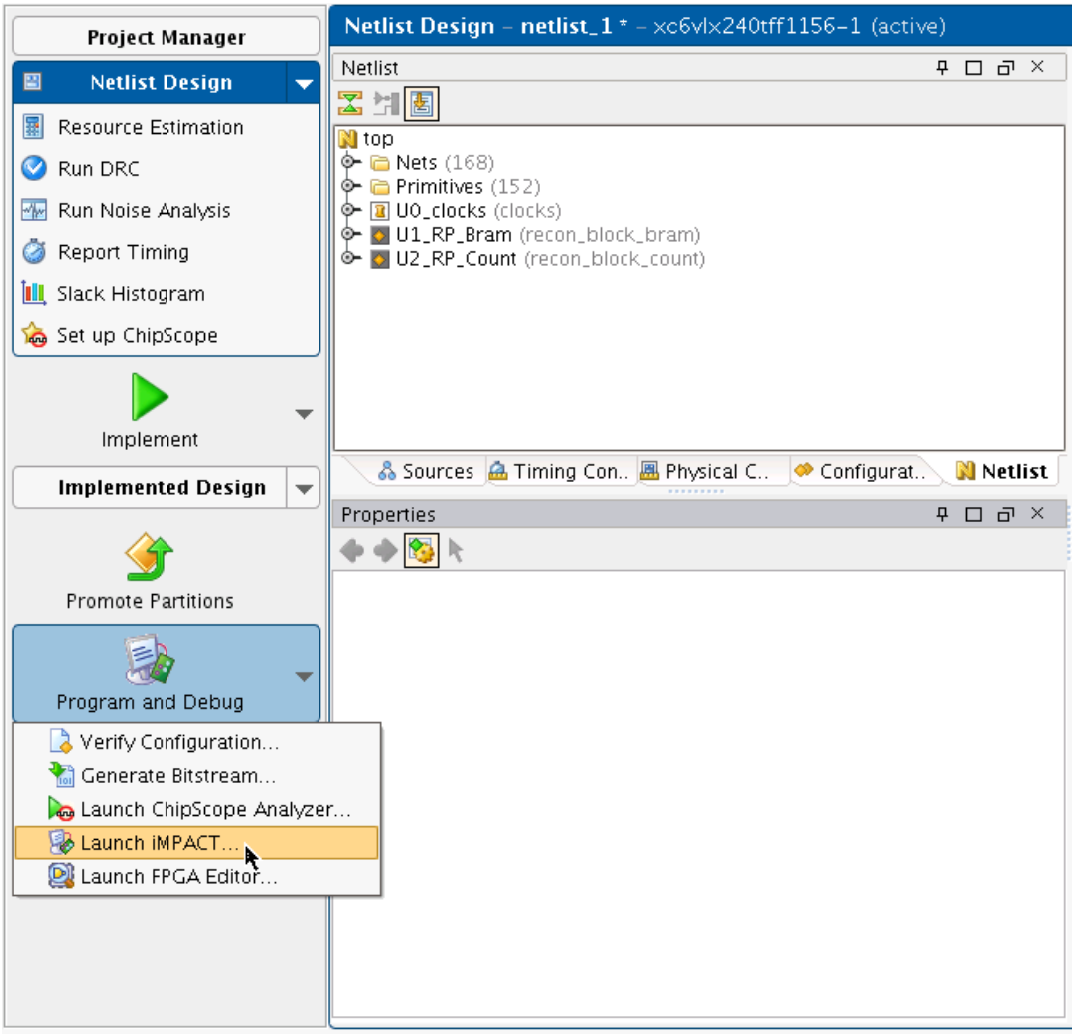

Figure 31: Launch iMPACT

12-1-6. In the iMPACT Flows frame double-click **Boundary Scan**, then click the Initialize Chain

button.

12-1-7. After the chain has been successfully detected, right-click the xc6vlx240t device and associate the full bit file `xpr_bram_led/PlanAhead/<project_name>/<project_name>.runs/config_1/config_1.bit`

12-1-8. Right-click xc6vlx240t again, and select **Program** (Figure 32).

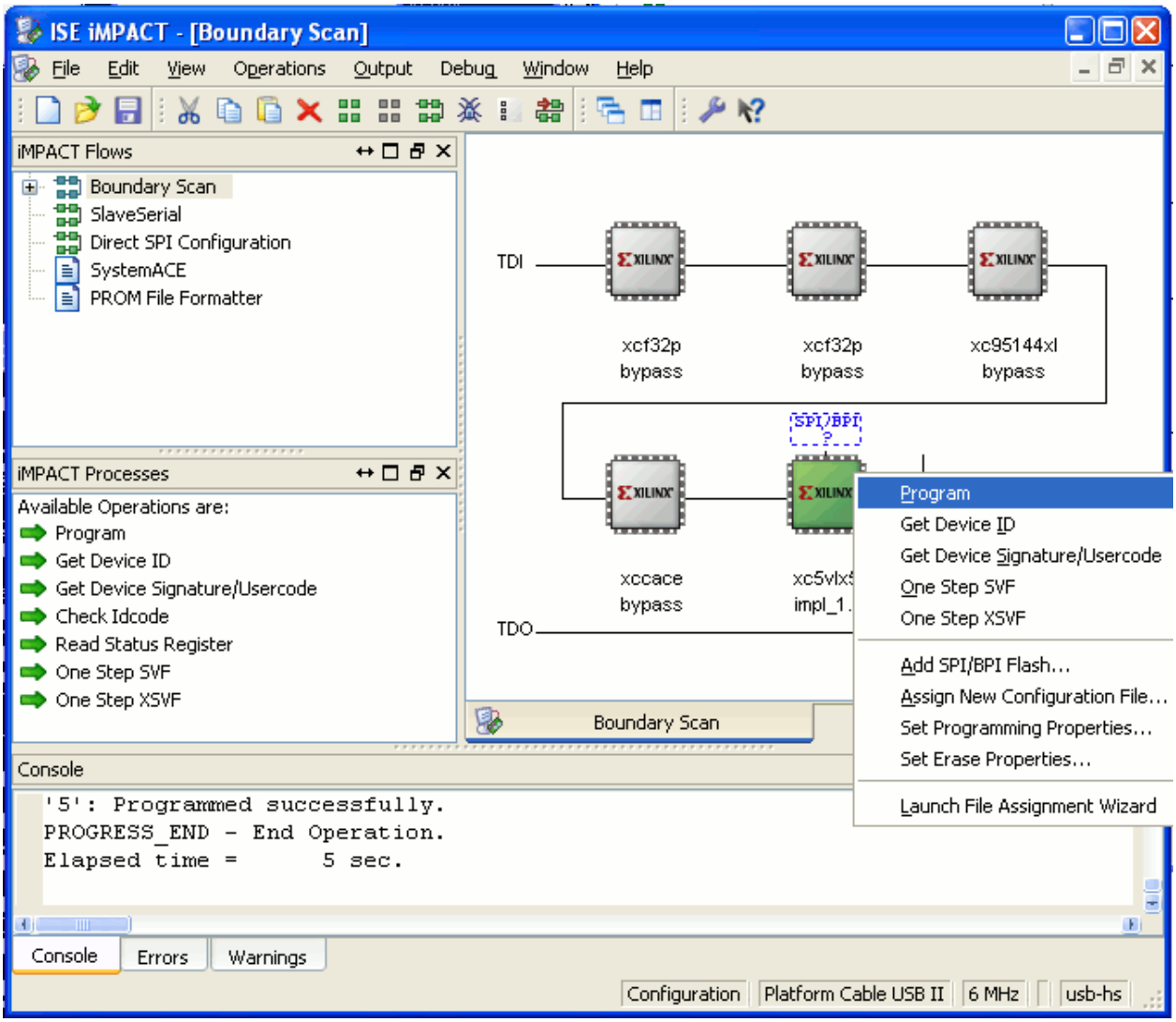

Figure 32: Configuring the Device through iMPACT

The FPGA on the ML605 will program with the full bit file. It will take several seconds to configure.

12-2. Associate a partial bit file.

12-2-1. Right-click the xc6vlx240t device, and select the partial bit file: `xpr_bram_led/PlanAhead/<project_name>/<project_name>.runs/config_2/config_2_U1_RP_Bram_BramSecond_partial.bit`.

12-2-2. Right-click the xc6vlx240t device, and select **Program**.

Notice that partially reconfiguring the FPGA is practically instantaneous because the partial bit file is very small.

Conclusion

In this tutorial, you created a PR PlanAhead project. You then created two Reconfigurable Partitions (RPs) and associated multiple Reconfigurable Modules to each RP. Each RP was then constrained to an area of the device using AREA_GROUP constraints, and you created global timing constraints to constrain the entire design. An initial configuration was implemented and promoted, and then you created additional configurations that imported the static logic from the initial configuration. PR Verify was run to verify the cohesiveness of all the configurations, and bit files were generated. Finally you downloaded a full bit file to the ML605 board containing the BramFirst and CountCW modules, and then reconfigured the U1_RP_Bram RP with a partial bit file for the BramSecond module.