Partial Reconfiguration of a Processor Peripheral Tutorial

PlanAhead Design Tool

UG744 (v 13.4) January 18, 2012

XILINX®

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
07/06/2011	13.2	Revalidated for the 13.2 release. Editorial updates only; no technical content updates.
10/19/2011	13.3	Updated the tutorial to use an AXI4-based design.
1/18/2012	13.4	Revalidated for the 13.4 release. Editorial updates only; no technical content updates.

Table of Contents

Partial Reconfiguration of a Processor Peripheral Tutorial

Introduction

This tutorial shows you how to develop a partial reconfiguration design using the Xilinx® Platform Studio (XPS), Software Development Kit (SDK), and the PlanAhead™ design tool. You will use XPS to create a processor hardware system which includes a lower-level module defining one Reconfigurable Partition (RP) and two Reconfigurable Modules (RMs). The two RM perform addition and multiplication functions. You will use SDK to create a software application which enables you to perform partial reconfiguration.

XPS and SDK are part of the Embedded Design Kit (EDK), which is included in the ISE® Design Suite Embedded and System Editions.

You will use PlanAhead to:

- Floorplan the design including defining a reconfigurable partition for the reconfigurable region
- Create multiple configurations and run the partial reconfiguration implementation flow to generate full and partial bitstreams.

You will use the ML-605 evaluation board to verify the design in hardware using a Compact Flash (CF) memory card to configure the FPGA device initially and then partially reconfigure the device using the AXI HWICAP peripheral by loading the partial bitstream files stored on the CF under the user software control.

This tutorial covers only a subset of the features contained in the PlanAhead tool bundled with ISE Design Suite Release. Other features are covered in other tutorials.

Tutorial Objectives

After completing this tutorial, you will be able to:

- Generate a processor system using XPS and SDK.
- Use the Partial Reconfiguration design flow capability in PlanAhead to generate full- and partial-bitstreams to dynamically reconfigure an FPGA design using the AXI HWICAP peripheral.

Getting Started

Software Requirements

The PlanAhead tool is installed with the ISE Design Suite 13.4 software. For this tutorial, you must have the Embedded or System edition of the ISE Design Suite installed. 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 13: Installation, and Licensing Guide* on the Xilinx website:

http://www.xilinx.com/support/documentation/sw_manuals/xilinx13.4/iil.pdf

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

Hardware Requirements

Xilinx recommends a minimum of 2 GB of RAM for use with this design for best performance.

Optionally, you can use an ML605 board and a USB download cable to test the hardware.

Locating the Tutorial Design Files

This tutorial uses a reference design, `UG744_design_files.zip`, which must be unzipped to a directory on your machine. Please note that the directory path you choose should not have a space in its name. You can download a copy of the reference design from the Xilinx website:

http://www.xilinx.com/support/documentation/dt_planahead_planahead13-4_tutorials.htm

Understanding the Processor System

This tutorial demonstrates how to implement a design that can be dynamically reconfigured using the AXI HWICAP peripheral.

The following figure shows a processor system. The design consists of a peripheral capable of performing a math function, having two unique capabilities: addition and multiplication.

You will verify the functionality with HyperTerminal under user application control. The dynamic modules are reconfigured using the AXI HWICAP peripheral.

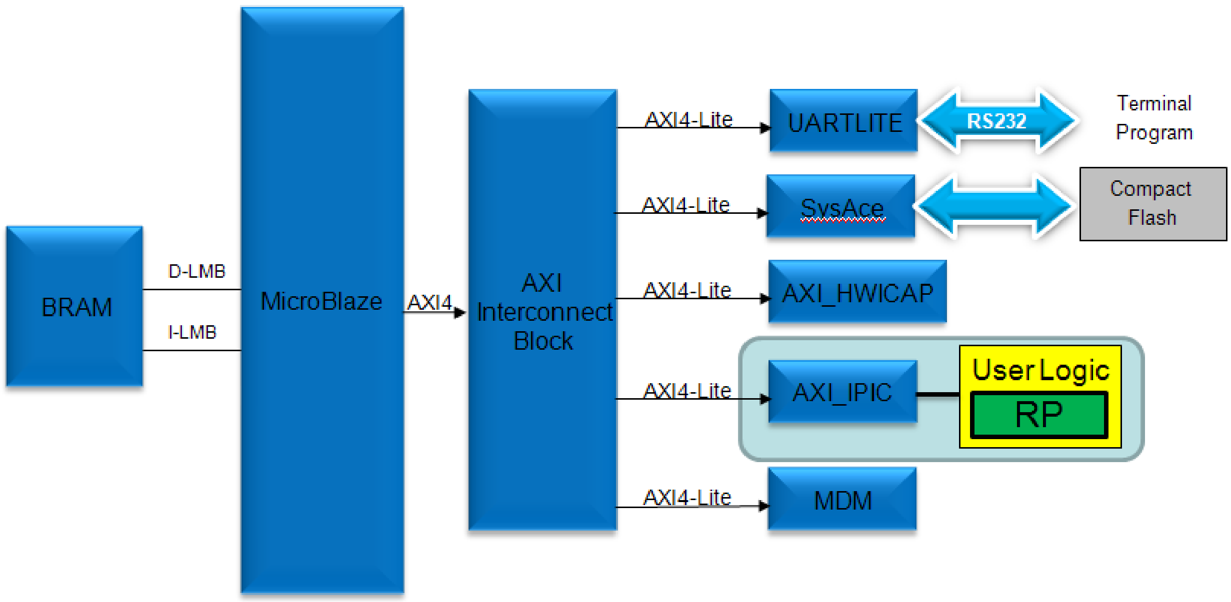

Figure 1: Top-Level Design

Project Directory Structure

The directory structure is:

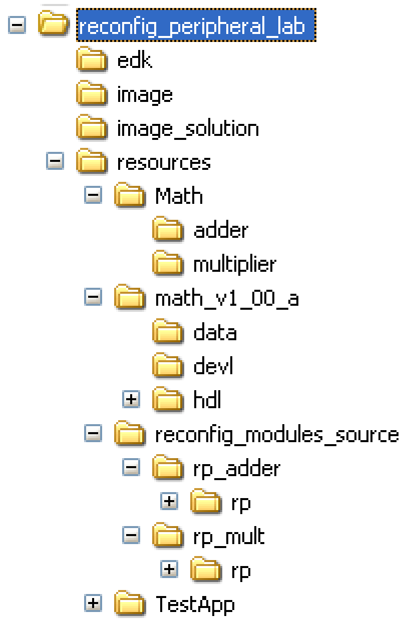

Figure 2: The Project Directory

- The `edk\` directory is used to create a processor system.
- The `resources\` directory contains:
    - o Source files used to generate the netlists of the addition and multiplication functions,
    - o A pre-compiled netlist for the addition and multiplication functions in Math and associate sub-directories, and
    - o A software application to demonstrate the functionality.
- The math processor core (pcore) that:
    - o Provides necessary processor bus connections
    - o Provides the required peripheral services (in this case, one slave register and a software reset)
    - o Is a placeholder for the math functionality module
- The `image\` directory is used to hold the generated full configuration bitstream file in the System ACE™ format and partial bitstream files.
- The `image_solution\` directory contains the final system.ace and partial bit files for a quick test.

Tutorial Steps

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 lab.

- Step 1: Creating a Processor Hardware System
- Step 2: Creating a Software Project
- Step 3: Creating a PlanAhead Project
- Step 4: Defining a Reconfigurable Partition
- Step 5: Adding Reconfigurable Modules
- Step 6: Defining the Reconfigurable Partition Region
- Step 7: Running the Design Rule Checker
- Step 8: Creating the First Configuration, Implementing, and Promoting
- Step 9: Creating Other Configurations, and Implementing
- Step 10: Running Partial Reconfiguration to Verify Utility
- Step 11: Generating Bit Files
- Step 12: Creating an Image, and Testing

Step 1: Creating a Processor Hardware System

Creating a Processor System Using the Base System Builder (BSB) Wizard in XPS

1. Select **Start > Programs > Xilinx Design Suite 13.4 > EDK > Xilinx Platform Studio** to open XPS.
2. In the Getting Started page, click **Create New Project Using Base System Builder** to open a **Create New XPS Project using BSB Wizard** dialog box.
3. Browse to the `reconfig_peripheral_lab\edk\` directory.
4. Click **Save**.
5. Keep the default options of using ISE tools and AXI System as the interconnect type, and click **OK**.

   You will create a system for a Virtex®-6 ML605 evaluation platform.
6. In Board and System Selection form, select **Xilinx** as a Board Vendor.
7. In Board Name field, select **Virtex-6 ML605 Evaluation Platform**.
8. In Board Revision field, select **D**.
9. Click **Next** with other default options selected.
10. Select **50.00 MHz** from the Processor Frequency drop-down menu.
11. Select **64 KB** from the Local Memory Size drop-down menu.
12. In the selected peripherals list on the right, remove all devices *except*:
    - RS232_*Uar*t_1
    - SysACE_CompactFlash
13. Click **RS232_Uart_1** and configure it with a baud rate of **115200**.
14. Click **Finish**.
15. If the Next Step dialog box opens, click **OK** to start using Platform Studio and open the **System Assembly View** window as shown in the following figure.

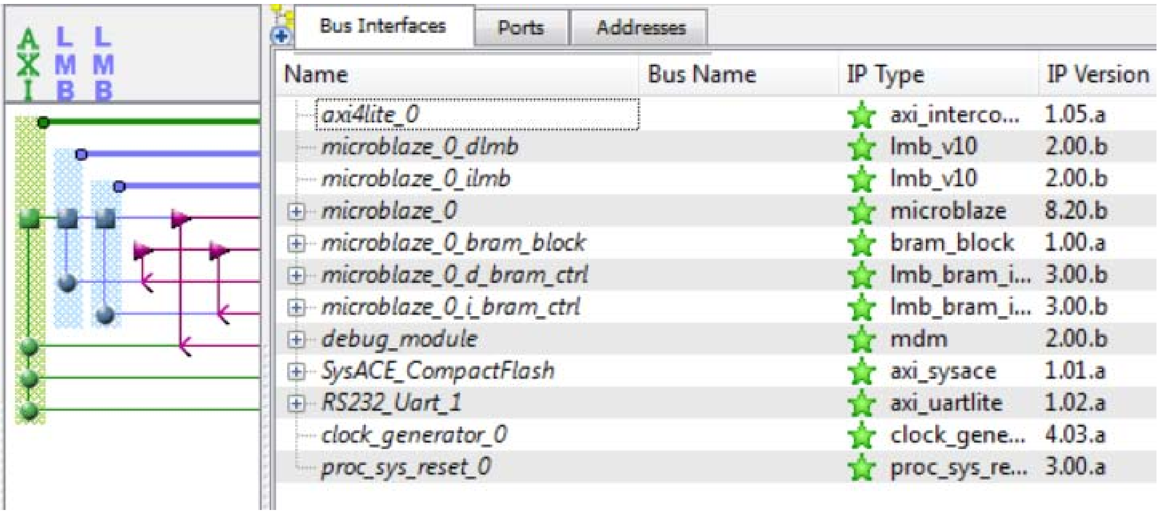

Figure 3: Displaying the System Assembly View

Adding the Required IPs to the Processor System

1. Copy the `reconfig_peripheral_lab\resources\math_v1_00_a\` folder to the `reconfig_peripheral_lab\edk\pcores\` folder.

   Partial Reconfiguration Design Details

   Examine the `user_logic.vhd` file located in `reconfig_peripheral_lab\resources\math_v1_00_a\hdl\vhdl\`. It declares a component that will be used in reconfigurable partition at line 133. The same is instantiated at line 158. The data inputs to the component are clocked at lines starting at 191. The reset input to the component is a combination of the hardware bus reset and software reset. The software reset is generated by a `soft_reset` block located at line 310 in `math.vhd` file located in the same directory. The software reset is necessary to reset the reconfigured logic after reconfiguring the partition.

   Note: If line numbering is hidden from view in XPS, turn line numbers on as follows:

   1. Select **Edit > Preferences > ISE Text Editor**.

   2. Click to select the **Show line numbers** check box.

   3. Click **Apply** and then **OK**.

2. Rescan the User Repositories in XPS by selecting **Project > Rescan User Repositories**.

   In the IP Catalog tab, MATH displays in the `USER` folder under the `Project Local pcores` folder.

3. Expand the `USER` folder.

4. Select **MATH**.

5. Double-click **MATH** to add an instance of the IP to the System Assembly.

6. A properties form will be displayed. Click **OK** twice to add the IP with the default settings and connect it to the `microblaze_0` instance.

7. In the IP Catalog tab, select the `FPGA Internal Configuration Access Port (v2.01.a)` IP (axi hwicap) under the `FPGA Reconfiguration` folder, right-click and select **Add IP**.

   This adds the instance of the IP to the System Assembly View.

8. Click **OK** twice to accept the default settings and connect the IP to the `microblaze_0` instance.

   Note that the IP cores are added, interface connections are made, and the addresses are automatically assigned.

Connecting the Ports

1. In the System Assembly View, select the **Ports** tab.
2. Expand the `axi_hwicap_0` instance.
3. Select **Hardware > Launch Clock Wizard**.
4. In the Clock Wizard form, select **clk_50_0000MHz** for the ICAP_Clk of the axi_hwicap_0 instance and select <AUTO> under the source column, and click **OK**.

   A warning message will appear.

5. Click **OK** to close the form.

   The connection appears as shown in Figure 4.

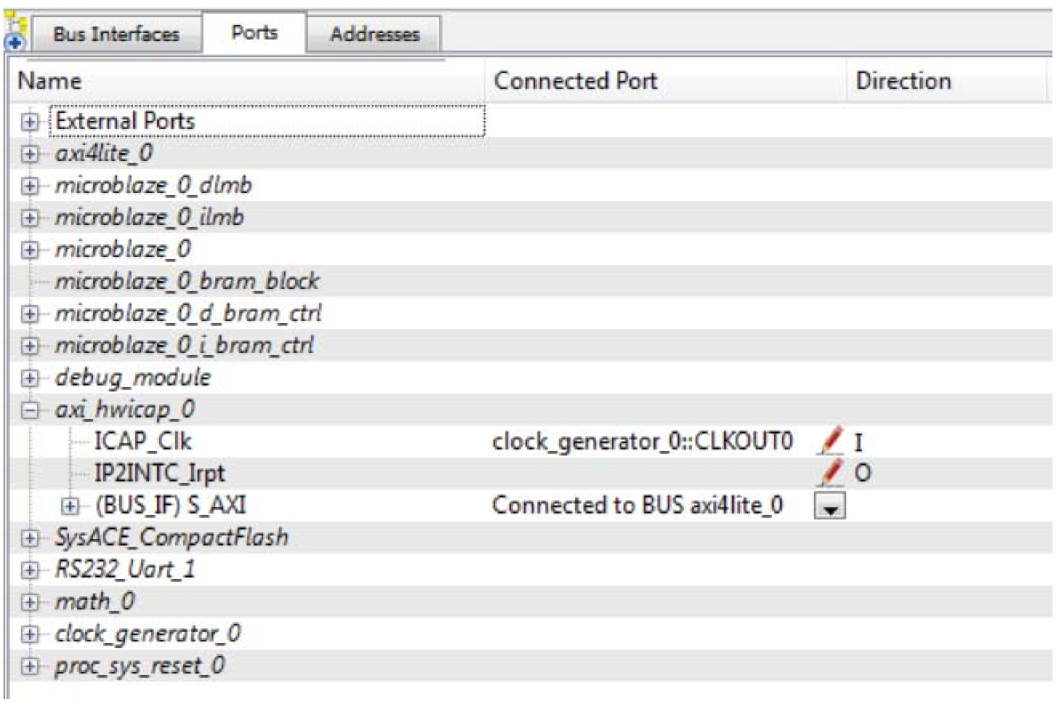

Figure 4: Connecting Clock Source to ICAP

Partial Reconfiguration Design Details

The `axi_hwicap pcore` allows separate clock domain for the `hwicap` so it can be run at 100 MHz when the system is run at a higher speed. In this tutorial, the system clock is 50.00 MHz and hence, we are running the entire design in a single clock domain.

Generating Netlists

1. To run the Platform Generator, select **Hardware > Generate Netlist**.

   This generates the peripheral and system netlists, and the `system.bmm` files, all of which are used during implementation in the PlanAhead tool.

Step 2: Creating a Software Project

Once the hardware netlist is generated, use the Software Development Kit (SDK) available with EDK to:

- Create a software project
- Import the provided source files
- Compile the provided source file
- Generate an executable file

Exporting Hardware Design to SDK, and Creating a Board Support Package

Be sure to add `xilfatfs` library support.

1. In XPS, select **Project > Export Hardware Design to SDK** to launch SDK.
2. Uncheck *Include bitstream and BMM file*.
3. Click **Export & Launch SDK**.

   A workspace location dialog box will appear.
4. Browse to the `reconfig_peripheral_lab\edk\SDK\SDK_Workspace` directory, and click **OK** to open SDK after importing hardware specification of the system.
5. In SDK, select **File > New > Xilinx Board Support Package**.

   Notice that the default Project Name is `Standalone_bsp_0` and the OS is `Standalone`.
6. Click **Finish** with default settings.

   The Board Support Package Settings window opens.
7. Check the **xilfatfs** check box to select the FAT file system support for the Compact Flash card.

	Name	Version	Description
☐	lwip140	1.00.a	lwIP TCP/IP Stack library: lwIP v1.4.0, Xilinx adapter v1.00.a
☑	xilfatfs	1.00.a	Provides read/write routines to access files stored on a FAT16/32 file system....
☐	xilflash	3.00.a	Xilinx Flash library for Intel/AMD CFI compliant parallel flash
☐	xilisf	2.04.a	Xilinx In-system and Serial Flash Library
☐	xilmfs	1.00.a	Xilinx Memory File System

Figure 5: Selecting File System Support

8. Click **OK** to accept the settings and close the form.

Creating a Xilinx C Project

1. Select **File > New > Xilinx C Project.**
2. Type **`TestApp`** for the Project Name.
3. Select **Empty Application** in the Project Application Template pane.
4. Click **Next**.

5. Select **Target an Existing Board Support Package**.
6. Click **Finish**.

Generating a Test Application

1. In the Project Explorer view, select **TestApp**.
2. Right-click and select **Import**.
3. Double-click **General**.
4. Double-click **File System**.
5. Browse to the `reconfig_peripheral_lab\resources\TestApp\src\` folder.
6. Click **OK**.
7. Select **main.c** and **xhwicap_parse.h**.
8. Click **Finish**.

   This compiles the source files and generates `TestApp.elf` in the `reconfig_peripheral_lab\edk\ TestApp\Debug\` folder.

   Partial Reconfiguration Design Details

   Examine the `reconfig_peripheral_lab\resources\TestApp\src\main.c` file.

   This code includes a function, beginning on line 164, which loads a partial bit file from the CompactFlash and writes to the ICAP.

   The calls to this function, beginning on line 433, instruct the program to load a specific partial bit file and then assert software reset.

   When the blank bitstream is loaded, the software reset is not required since there is no real logic residing in the reconfigurable region.

Generating a Linker Script

Be sure that the Heap and Stack sizes are set to 2048 (0x800).

1. In SDK Project Explorer view, select **TestApp**.
2. Right-click and select **Generate linker script**.
3. Change the Heap size and the Stack size to **2048**.

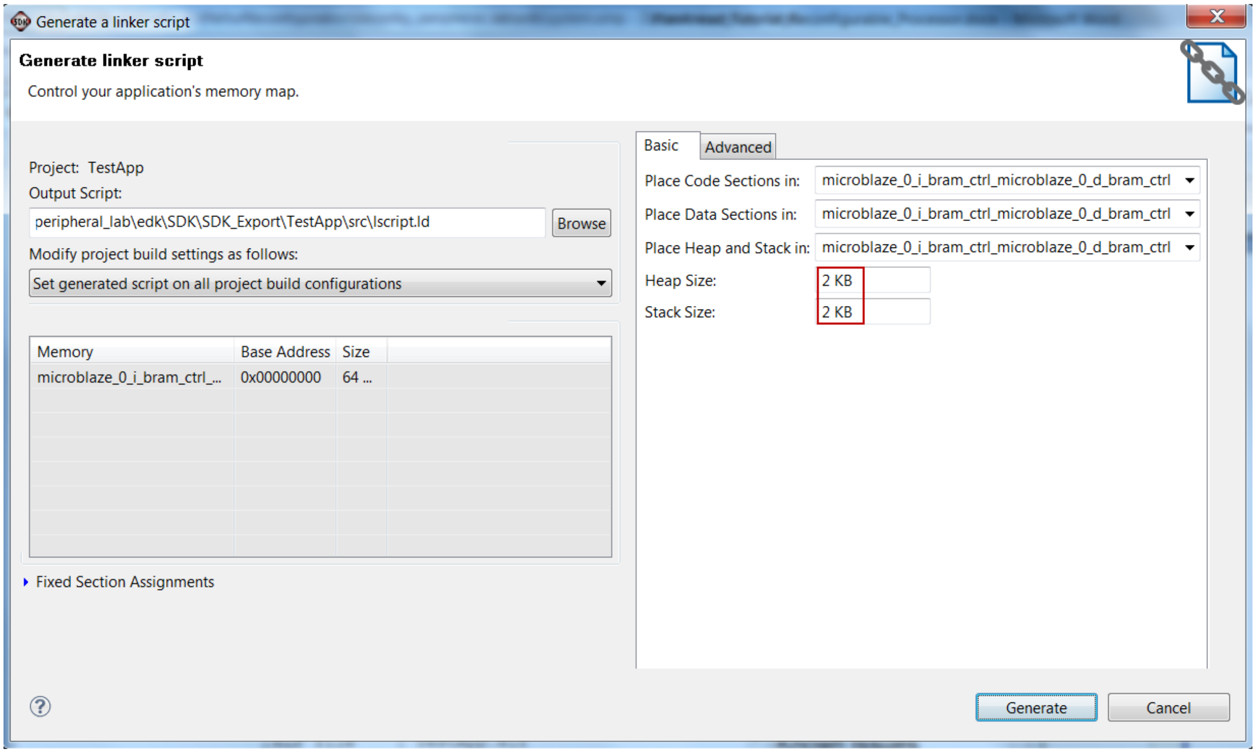

Figure 6: Generating a Linker Script

4. Click **Generate**.
5. Click **Yes** to overwrite the existing copy and recompile the application again.
6. Select **File > Exit** to close SDK.

Step 3: Creating a PlanAhead Project

Now that you have generated the required netlist files for the design, you will use the PlanAhead tool to:

- Floorplan the design
- Define reconfigurable partitions
- Add reconfigurable modules
- Run the implementation tools
- Generate full and partial bitstreams

In this step, you will create a new project.

Creating a PlanAhead Project, and Importing the Generated Netlist Files

1. To open PlanAhead, select **Start > Programs > Xilinx ISE Design Suite 13.4 > PlanAhead > PlanAhead**.
2. Click **Create New Project**.
3. Click **Next**.
4. Browse to and select the `reconfig_peripheral_lab\` directory for the Project location.
5. Click **Select**.
6. Type **`PlanAhead`** for the Project name in the New Project wizard.

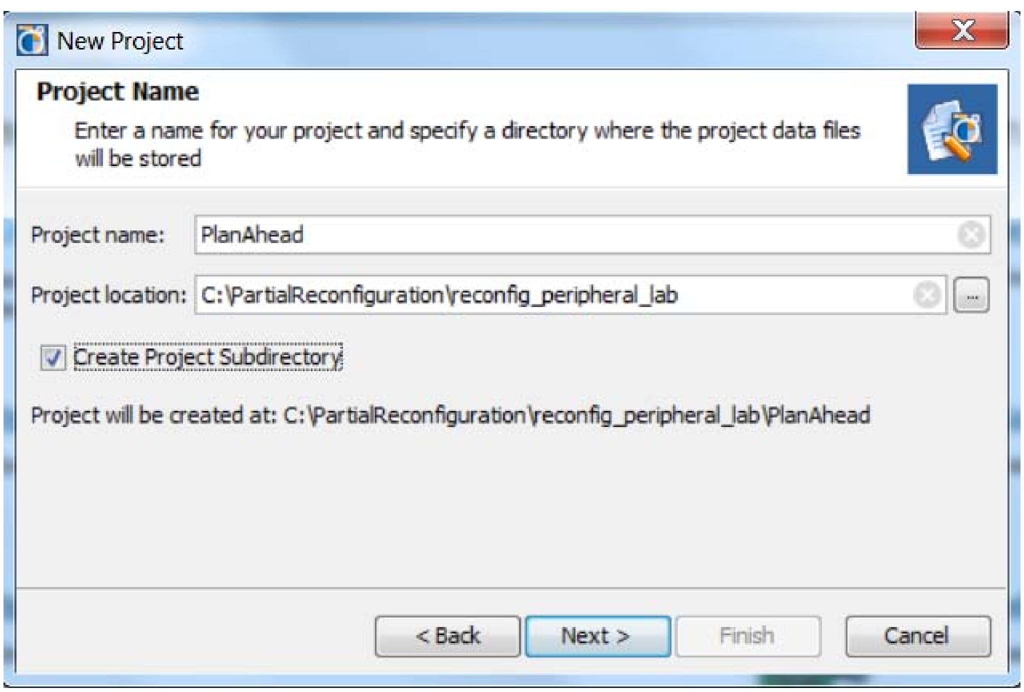

Figure 7: Project Name Page of the New Project Wizard

7. Click **Next**.
8. In the New Project Design Sources page, select **Specify synthesized (EDIF or NGC) netlist**.

9. Check the **Enable Partial Reconfiguration** option.

   Note: If you forget to check the option, you can still enable it from the project (netlist based only) by selecting **Tools > Project Settings > General** and clicking the "Partial Reconfiguration Project" check box. This must be done before a partition can be defined as reconfigurable.

10. Click **Next**.

    Note: The Enable Partial Reconfiguration option is available only if you have a license for Partial Reconfiguration.

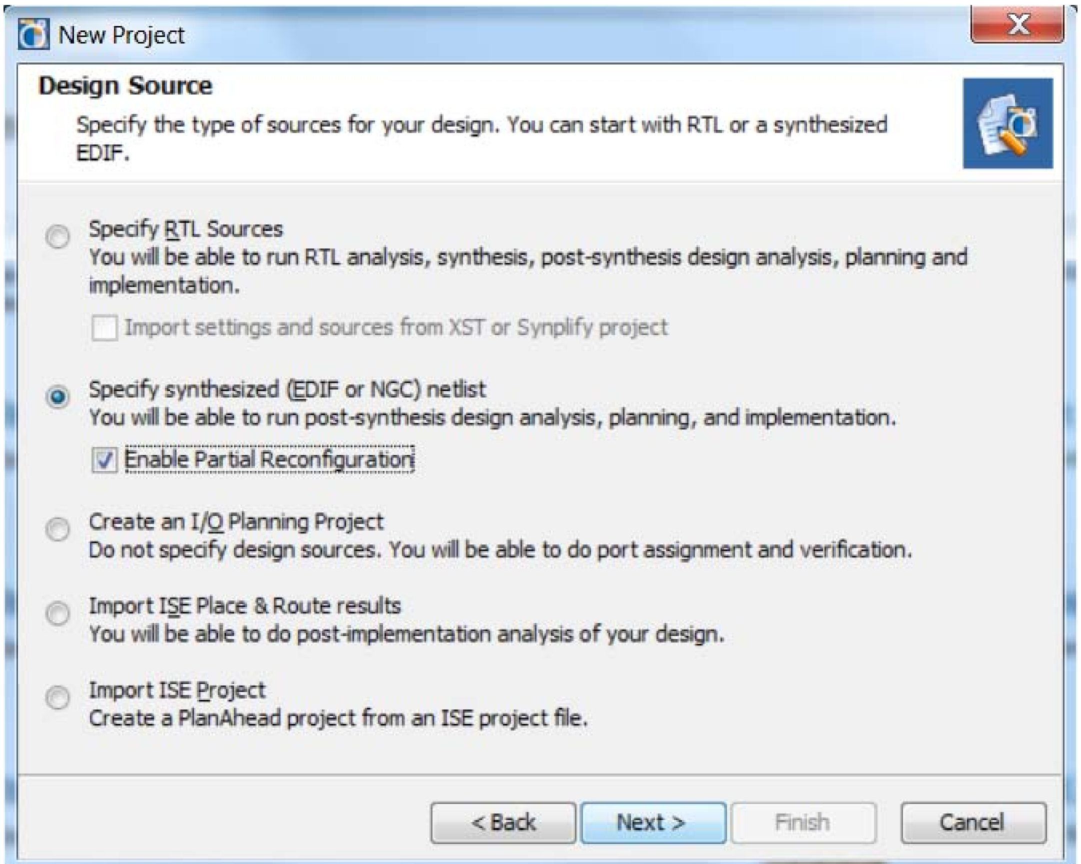

Figure 8: Importing Synthesized Netlists

11. Click the **Add Files** button.

12. Browse to `reconfig_peripheral_lab\edk\implementation\`

13. Select all NGC files including the `system.ngc` file, and click **OK**.

14. In the Top column, click the radio button next to `system.ngc` to identify it as the top-level design file.

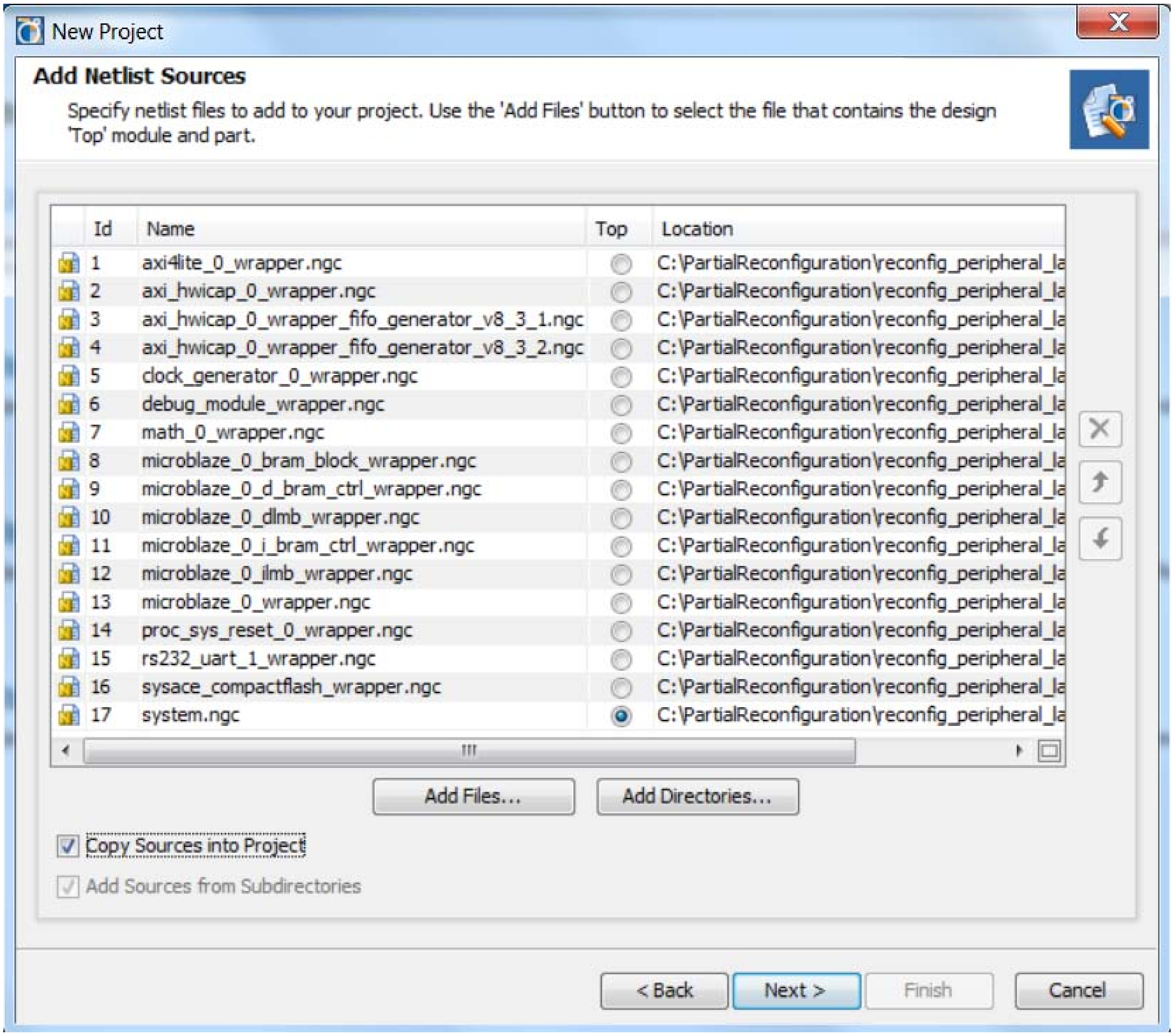

Figure 9: Selecting the Top-level Netlist File

15. Click **Next**.

    The Add Constraint files (optional) page opens.

16. Click **Add Files**.
17. Browse to `reconfig_peripheral_lab\edk\data\`
18. Select **system.ucf**.
19. Click **OK**.
20. Click **Next** to open the Product Family and Default Part page.
21. Make sure that the **xc6vlx240tff1156-1** part is selected. Otherwise, select the filters, and select the **xc6vlx240tff1156-1** part as shown in the following figure.

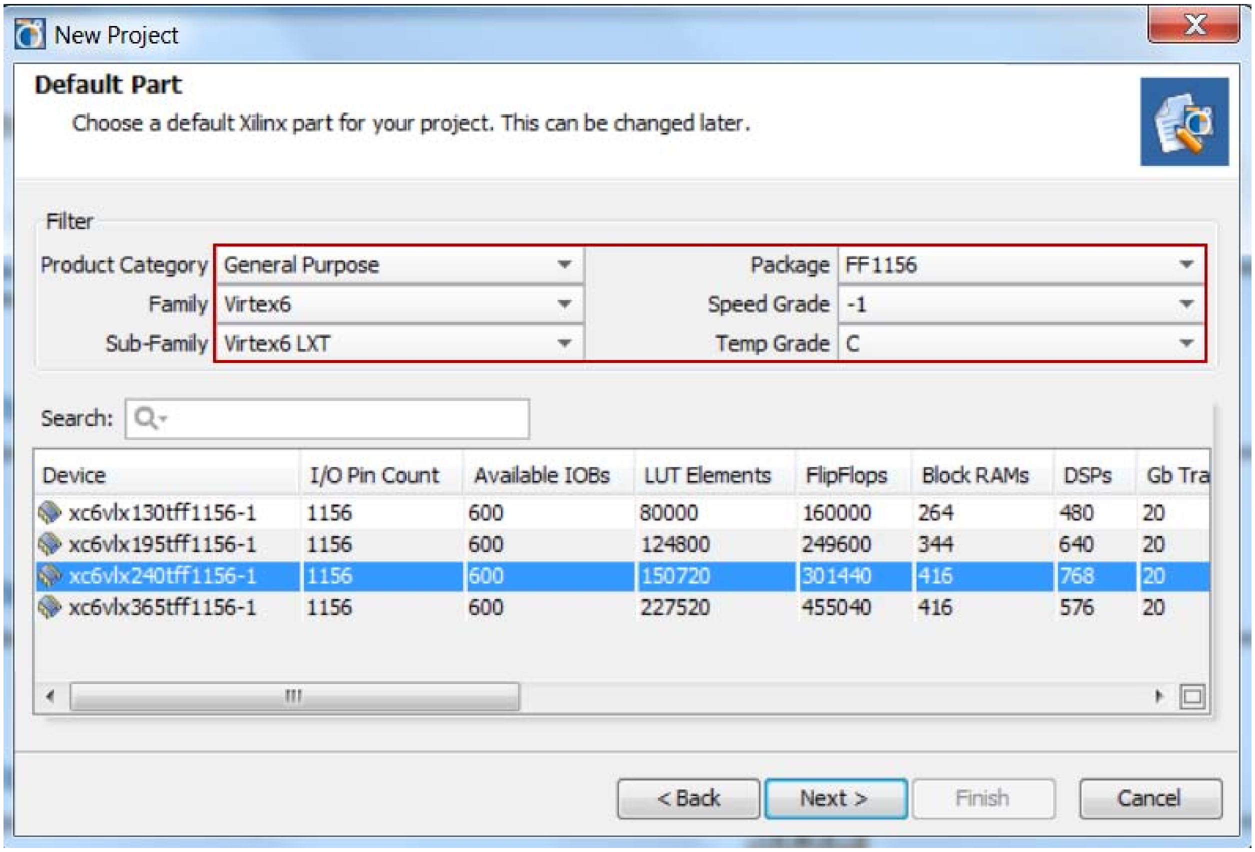

Figure 10: Selecting the Target Device

22. Click **Next**.
23. Click **Finish**.

    The project is created. The Project Manager pane displays the modules present in the design.

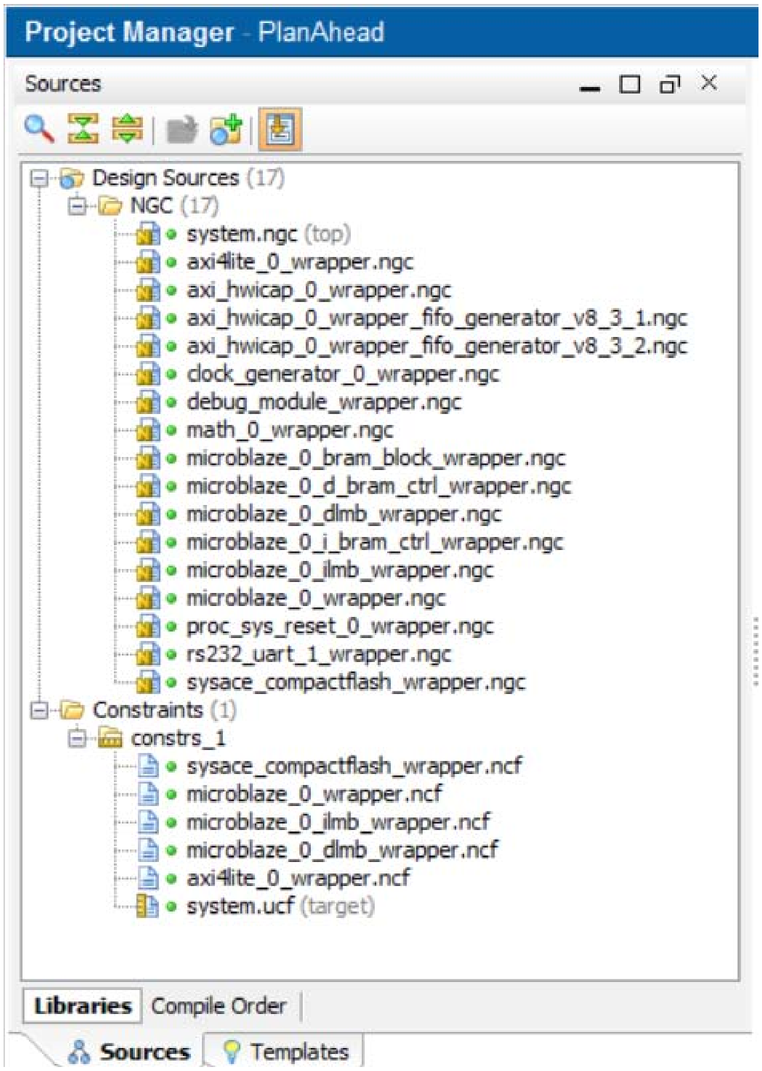

Figure 11: Design Hierarchy in PlanAhead

Step 4: Defining a Reconfigurable Partition

This design has one reconfigurable partition that must explicitly be defined.

Defining a Reconfigurable Partition (RP) With a Black Box Reconfigurable Module (RM).

1. Click **Netlist Design** to invoke the netlist files parser.

   This is necessary as we want to access a lower-level module to define a reconfigurable partition.

   A warning message indicating that one instance will be converted to a black box because the netlist file for it is missing. This is expected because no netlist has been associated with this module yet.

   A Netlist tab displays the hierarchical view of the system.

2. Click **OK**.

3. Expand the `math_0` instance.

4. Select **math_0/USER_LOGIC_I/rp_instance** in the Netlist view.

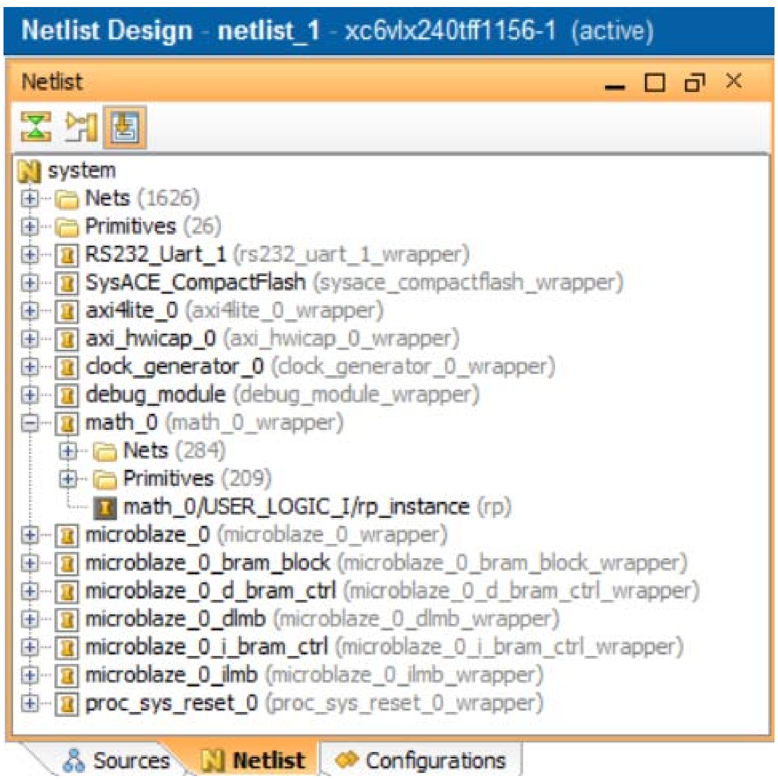

Figure 12: Netlists Hierarchy View

5. Right-click and select **Set Partition**.

6. Click **Next** *twice*.

   The Set Partition dialog box will appear.

7. Select **Add this Reconfigurable module as a black box without a netlist**.

8. Type **math_BB** in the RM name field since the partition does not yet have a defined netlist.

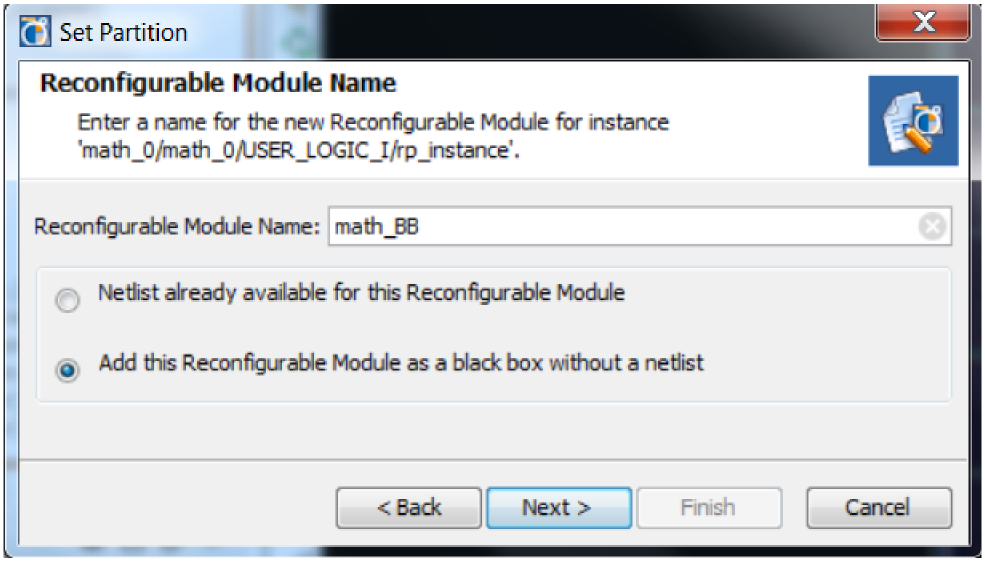

Figure 13: Setting a Partition

9. Click **Next**.
10. Click **Finish**.

Note: The black box icon has changed to a diamond shape.

Step 5: Adding Reconfigurable Modules

This design has two Reconfigurable Modules (RMs) for the Reconfigurable Partition (RP). In this step, you will add the two modules.

Adding Two Reconfigurable Modules: Adder and Multiplier

1. In the Netlist window, select the `math_0/USER_LOGIC_I/rp_instance`.
2. Right-click and select **Add Reconfigurable Module**.
3. Click **Next**.

   The Add Reconfigurable Module dialog box displays.
4. In the Reconfigurable Module Name field, type **`adder`**.
5. Verify that **Netlist already available for this Reconfigurable Module** is selected.

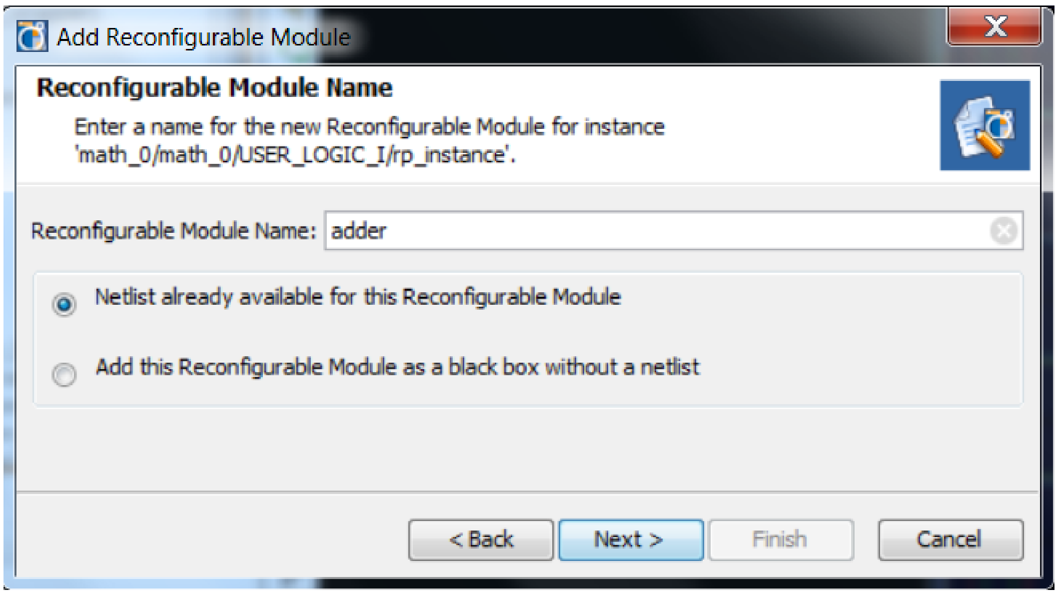

Figure 14: Adding a Reconfigurable Module

7. Click **Next**.
8. Browse to `reconfig_peripheral_lab\resources\Math\adder\` and select the **rp.ngc** file.
9. Click **Open**.

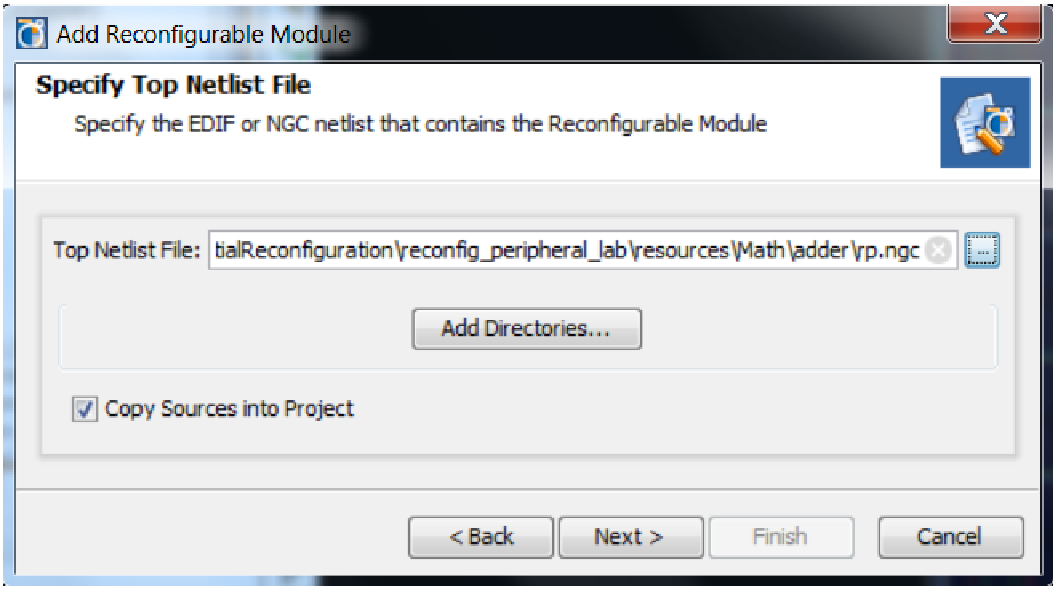

Figure 15: Locating the adder Version of math.ngc

10. Click **Next** *twice*.
11. Click **Finish**.
12. In the Netlist pane, expand `Reconfigurable Modules` hierarchy under `math_0/USER_LOGIC_I/rp_instance` to view the adder RM entry.
13. Follow the steps in Step 5 to add a **multiplier** RM from the `reconfig_peripheral_lab\resource\Math\multiplier\rp.ngc` directory. Name the RM **mult**.

    The Netlist window displays three Reconfigurable Modules (including the black box) for the math Reconfigurable Partition.

    The multiplier module is active (with a check mark) as it was the most recent netlist to be added to the project.

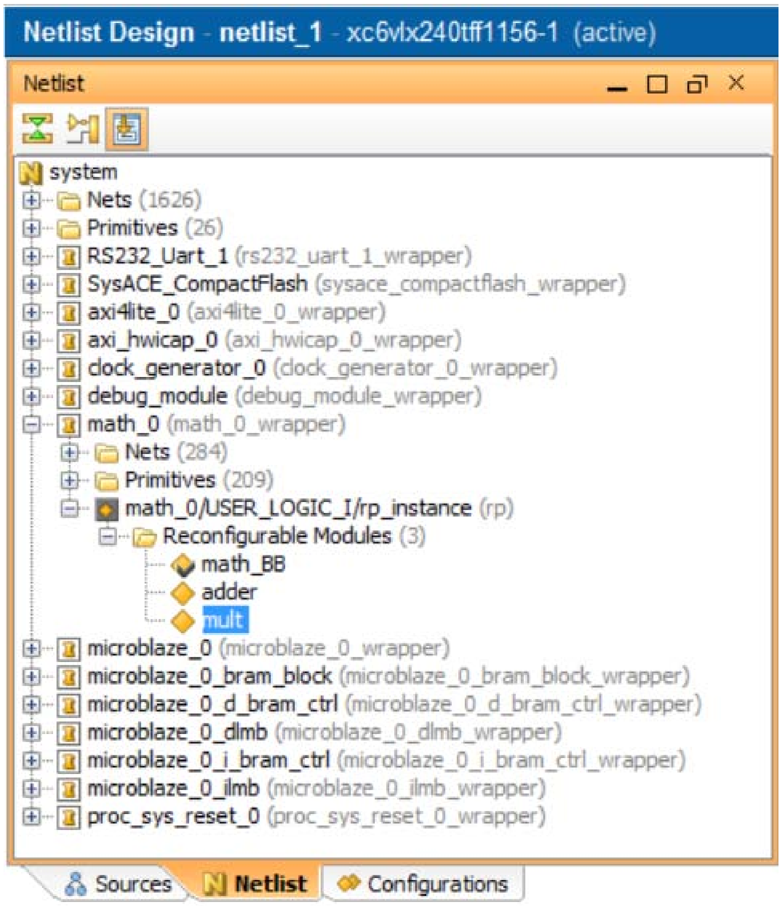

Figure 16: PlanAhead Project with adder and mult RMs Added

Step 6: Defining the Reconfigurable Partition Region

Next, floorplan the RP region. Depending on the type and amount of resources used by each RM, the RP region must be appropriately defined so it can accommodate any RM variant.

Setting the Reconfigurable Region

1. Select **Window > Physical Constraints**.
2. In the Physical Constraints tab, select **pblock_math_0/USER_LOGIC_I/rp_instance**.
3. Right-click and select **Set Pblock Size**.

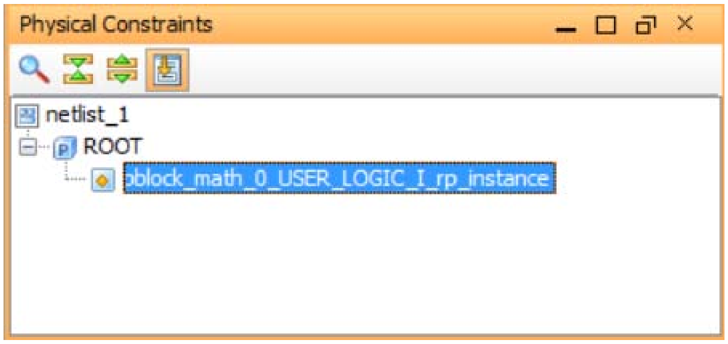

Figure 17: Setting Physical Constraints

4. Zoom to the top left quarter of the FPGA.
5. Move the cursor in the Device window.
6. Click and drag the cursor to draw a box that bounds SLICE_X8Y230:SLICE_X17Y239, as shown below.

   Drawing a box around this region is required because the multiplier (mult) RM requires one DSP48E and the adder RM requires 32-bit tall carry chain.

   The current grid coordinates are reported in the status bar at the bottom of the PlanAhead window.

   At the completion of this step, the Set Pblock dialog box displays.

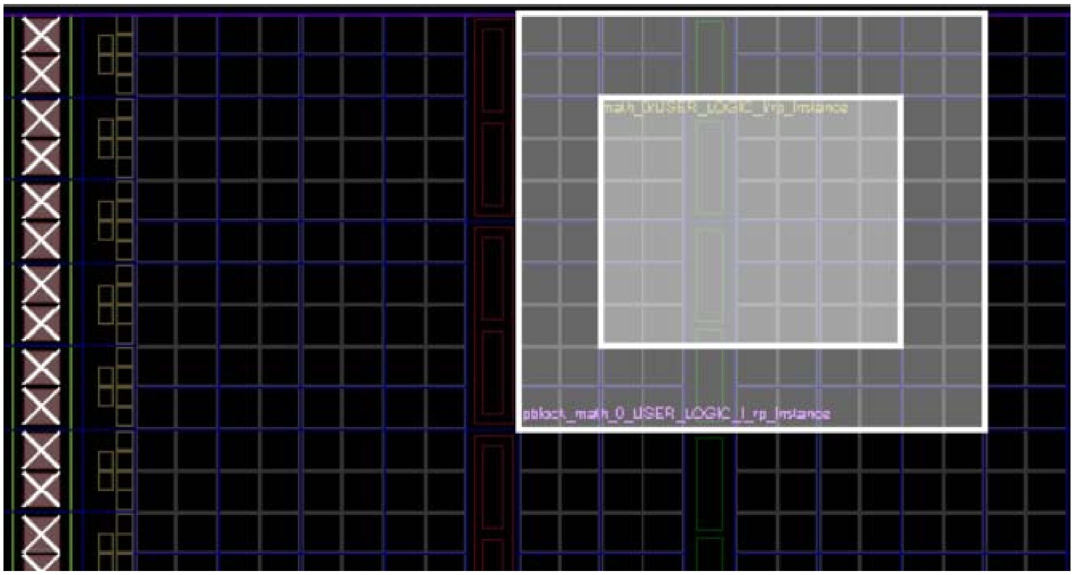

Figure 18: Closer View of the Pblock Area

7. In the Set Pblock dialog box, verify that **SLICE** and **DSP48** are checked as the resources to be reconfigured, shown in the following figure.
8. Click **OK**.

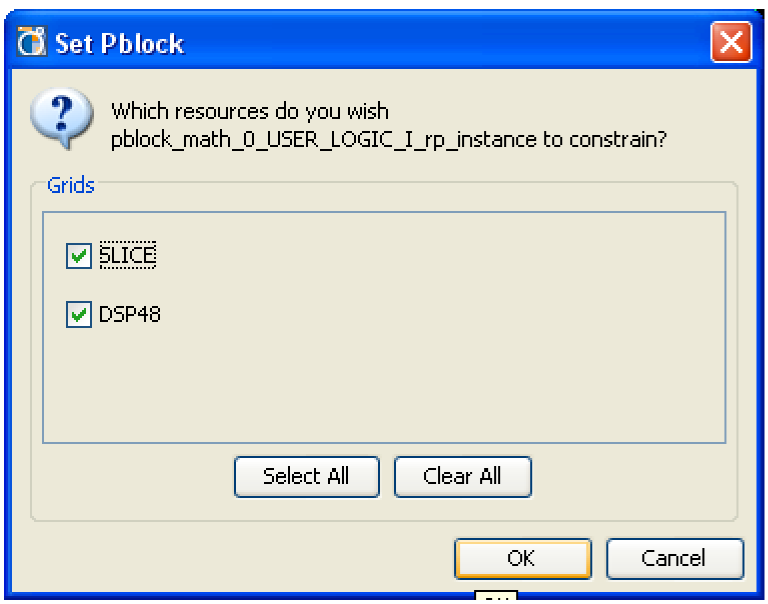

Figure 19: Setting Pblock with SLICE and DSP48

Step 7: Running the Design Rule Checker

Xilinx recommends that you run a Design Rule Check (DRC) in order to detect errors as soon as possible.

Selecting and Running PR-specific DRCs

1. Select **Tools > Run DRC**.
2. Deselect **All Rules**.
3. Select **Partial Reconfig**.
4. Click **OK** to run the PR-specific design rules.

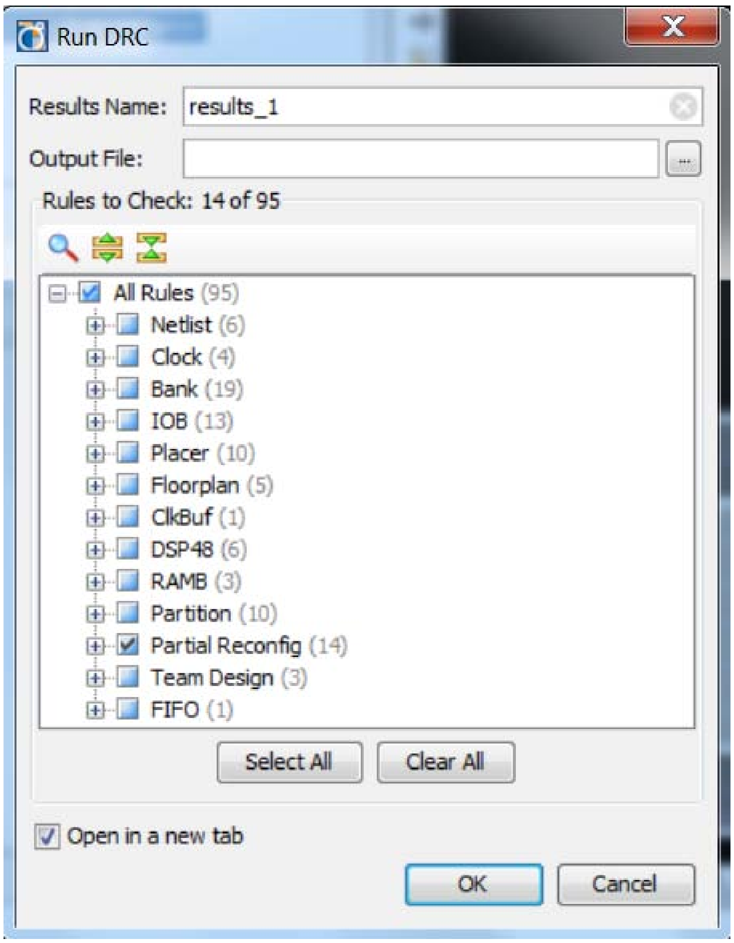

Figure 20: Running Design Rule Checks

You will see warnings stating that Reconfigurable Modules (RMs) have not been implemented.

Step 8: Creating the First Configuration, Implementing, and Promoting

Now you can create and implement the first configuration.

Creating a New Strategy

Use the `-bm` option pointing to the `system.bmm` file for the new strategy.

1. Select **Tools > Options**.
2. Select **Strategies** in the left pane.
3. Select **ISE 13** in the **Flow** drop-down box.
4. Under PlanAhead Strategies, select **ISE Defaults**.
5. Click the + button to create a new strategy.

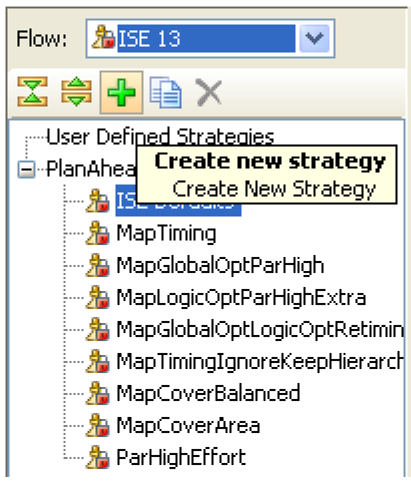

Figure 21: Creating a New Strategy

6. Name the new strategy **`ISE13_BM`**.
7. Click **OK**.
8. Under **Translate (ngdbuild)**, click in the **More Options** field.
9. Type **`-bm ..\..\..\edk\implementation\system.bmm`**, and click **Apply**.

Running the Implementation Using Mult as a Variant

1. At the bottom of the PlanAhead tool user interface, select the **Design Runs** tab.
2. Select the **config_1** run.
3. In the Implementation Run Properties window, select the **General** tab.
4. In the Name field, type **`mult`** as the run name.
5. Click **Apply** to change the run name from `config_1` to `mult`.

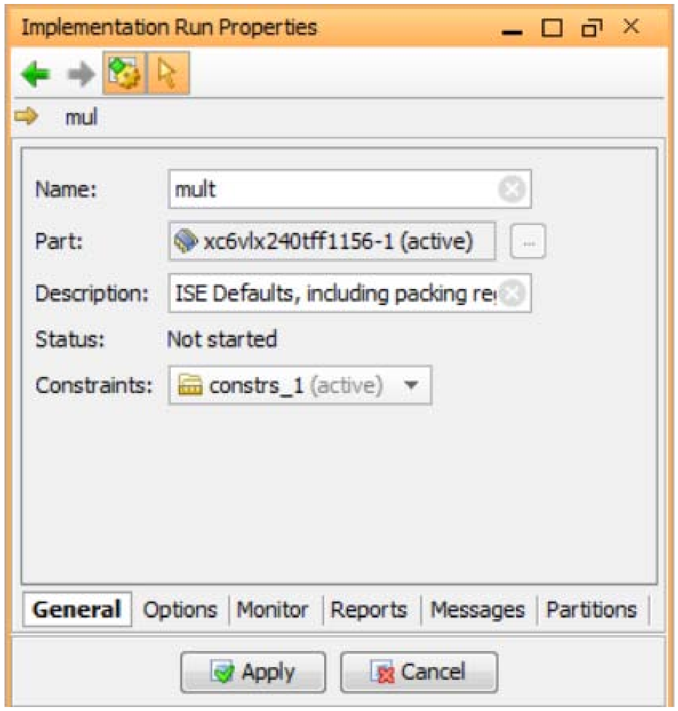

Figure 22: Implementation Run Properties View

6. In the Options tab, change the Strategy to **ISE13_BM**.
7. Click **Apply**.
8. In the Partitions tab, click the Module Variant column drop-down button and select **mult** as the variant.
9. Click **Apply**.

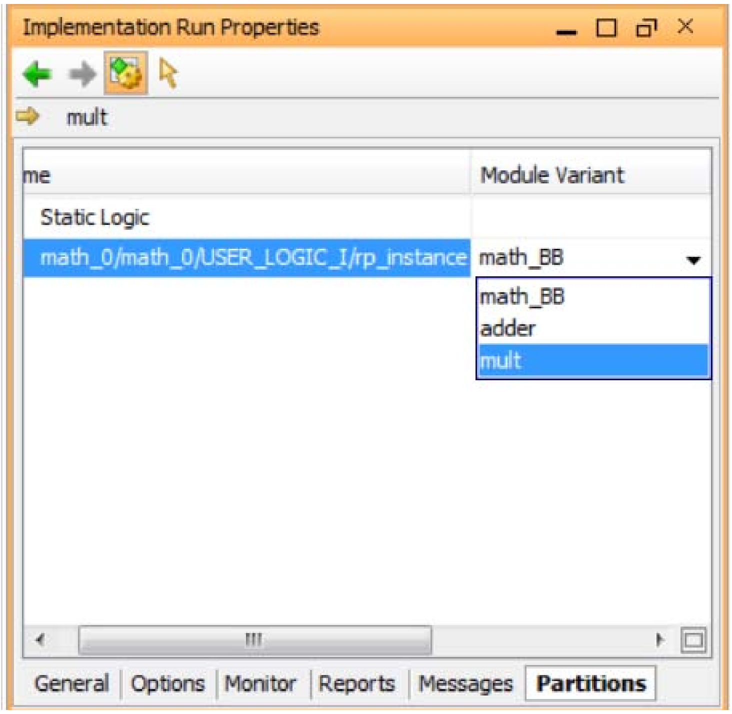

Figure 23: Implementation Run

10. In the Design Run window, select **mult**, and right-click and select **Launch Runs** to run the implementation.
11. Select **Launch Runs on Local Host**.
12. Click **OK**.
13. Click **Save** to save the project and run the implementation.

    The implementation runs.

    When implementation is finished running, a dialog box opens in which you can load the implemented results, or promote the implemented partitions, among other options.
14. Select the **Promote Partitions** radio button, and click **OK**.
15. In the Promote Partitions dialog box, click **OK** to promote the current configuration so the implemented results are available for the subsequent configurations.

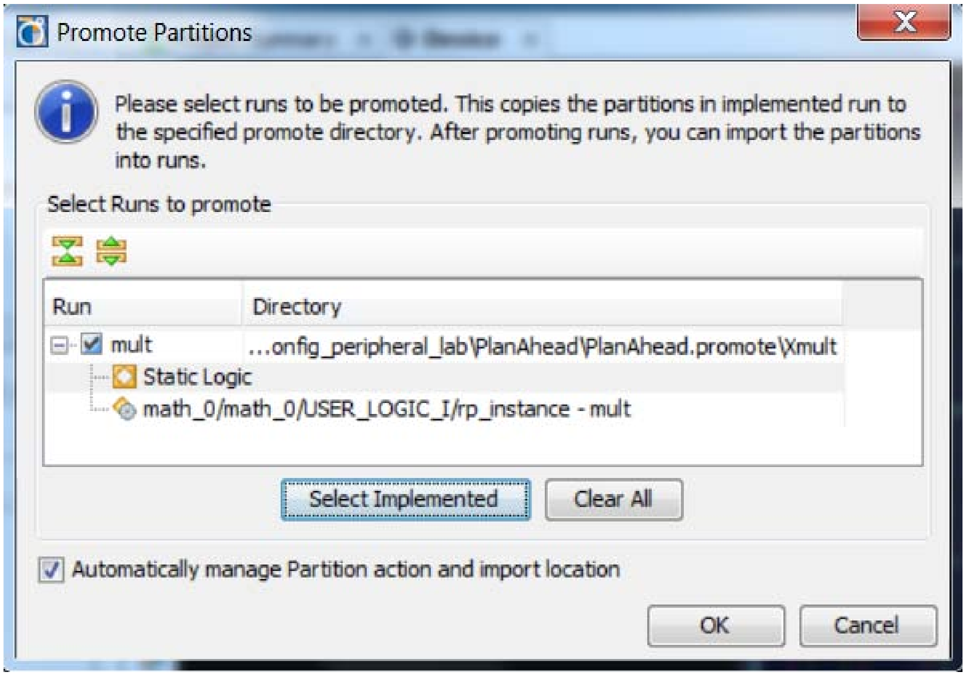

Figure 24: Promoting Partitions

Step 9: Creating Other Configurations, and Implementing

After you have created the first configuration, the static logic implementation is reused for the rest of the configurations. Next, you will create the desired number of additional configurations and implement them.

Creating Multiple Runs

1. Select **Flow > Create New Runs**.

   The Create Multiple Runs window opens.

2. Click **Next** *twice*.
3. In the Choose Implementation Strategies and Reconfigurable Modules page, change the name of the configuration from `config_1` to **`adder`**.
4. Click **More**.
5. Change the name of `config_1` to **`black_box`**.

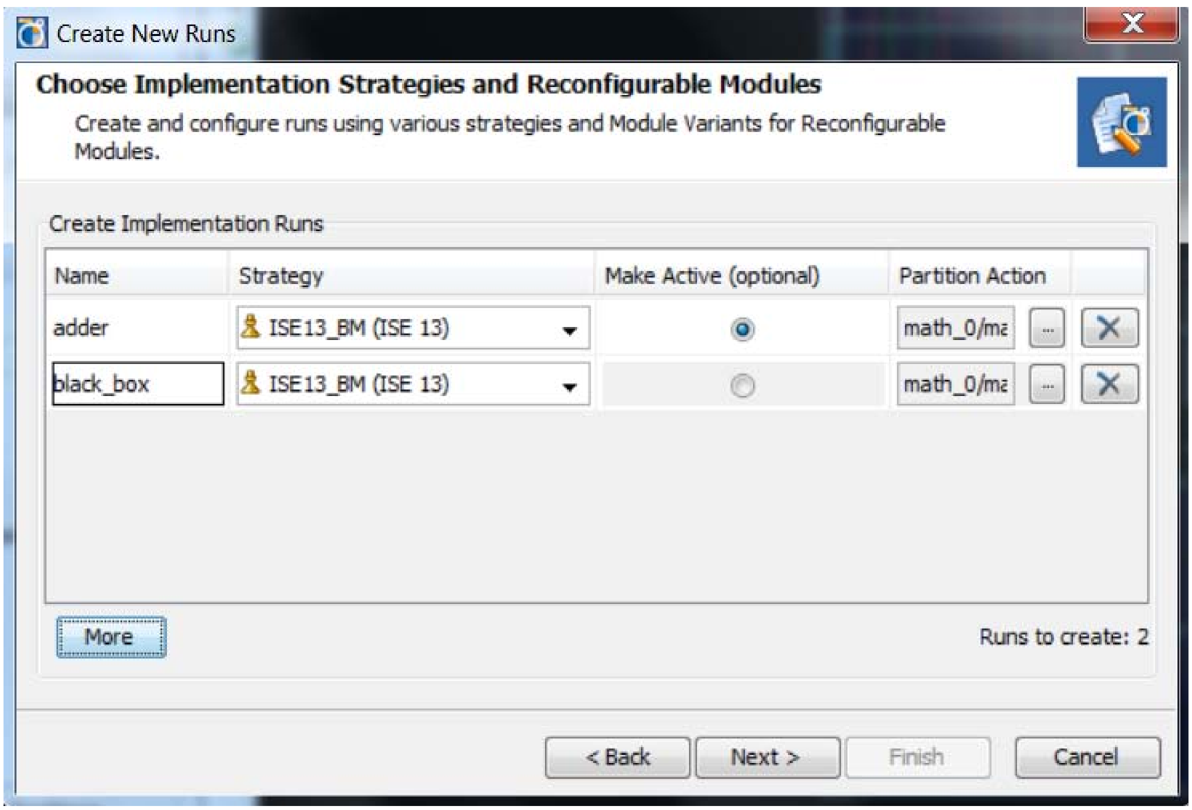

Figure 25: Creating Multiple Runs

6. In the adder configuration row, click the **Partition Action** field.
7. For the rp_instance row, click the Module Variant column drop-down arrow, and select **adder** as the variant to be implemented, as shown in Figure 26.

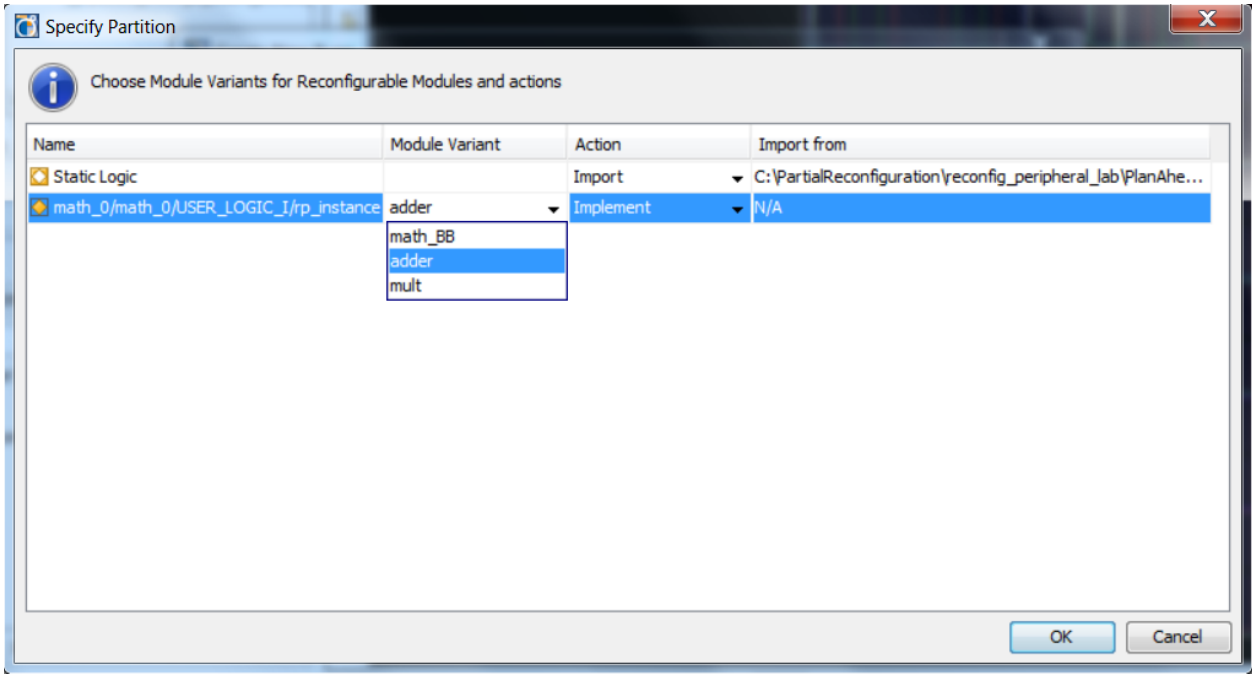

Figure 26: Selecting adder Module Variant

8. Click **OK**.
9. Similarly, select **math_BB** variant for the black_box run (row).
10. Click **Next**.
11. Select **Launch Runs on Local Host**.
12. Click **Next**.
13. Click **Finish** to run the implementations for both configurations.
14. Click **Cancel** when the runs are finished.

Step 10: Running Partial Reconfiguration to Verify Utility

Next, you will check to be sure that the static implementation, including interfaces to reconfigurable regions, is consistent across all configurations. To verify this, you can run the PR_Verify utility.

Running the PR_Verify Utility

Run the PR_Verify utility to make sure that there are no errors.

1. In the Configurations window, select any of the configurations.
2. Right-click, and select **Verify Configuration**.

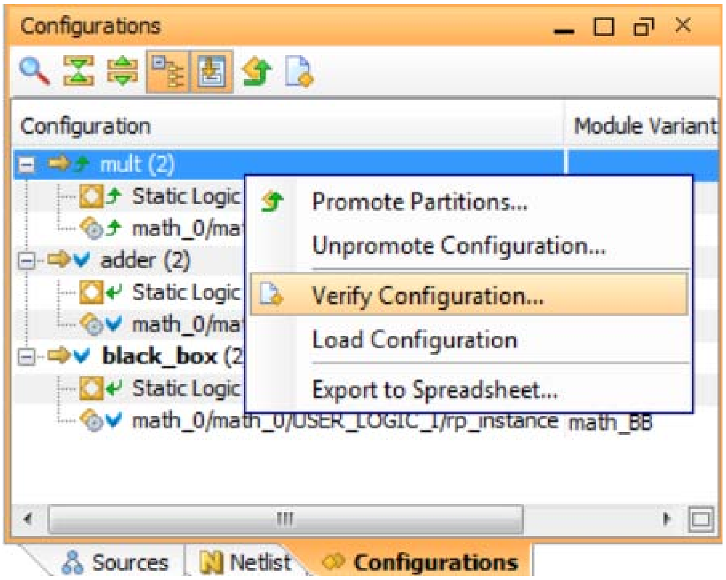

Figure 27: Verifying All Configurations

3. Press **Shift** and select all configurations.
4. Click **OK**.
5. The PR_Verify utility runs and reports that there were no errors.

Step 11: Generating Bit Files

After all the configurations have been validated by PR_Verify, you can generate full and partial bit files for the entire project.

Generating Full and Partial Bitstreams

1. In the Design Runs window, press **Shift** and select the following three designs runs:
   - **mult**
   - **adder**
   - **black_box**
2. Right-click and select **Generate Bitstream**.

   This runs the bitstream generation process and generates full and partial bitstreams.

   The bit files are placed in the `mult`, `adder` and `black box` directories under the `reconfig_peripheral_lab\PlanAhead\PlanAhead.runs\` directory.
3. Click **OK**.
4. Save the project.
5. Close PlanAhead.

Step 12: Creating an Image, and Testing

For this step you need to open an EDK shell, and create both a `download.bit` and a `system.ace` file in the `image\` directory. Copy the generated partial bit files, place them in the `image\` directory, and name them `adder.bit`, `mult.bit`, and `blank.bit`.

Renaming Partial Bitstream Files, and Generating the system.ace File

1. Launch the EDK bash shell or ISE Design Suite command prompt as follows:
   - From XPS, select **Project > Launch Xilinx Shell**, or
   - From your Windows environment, select **Start > Programs > Xilinx ISE Design Suite 13.4 > Accessories > ISE Design Suite Command Prompt**.
2. In the Xilinx shell or command window, go to the `reconfig_peripheral_lab\image\` directory.
3. Execute the following command to generate the `download.bit` file (with the software component included) from `adder.bit` (with the hardware component) only.

   ```
   data2mem -bm ..\edk\implementation\system_bd
   -bt ..\PlanAhead\PlanAhead.runs\adder\adder.bit
   -bd ..\edk\SDK\SDK_Export\TestApp\Debug\TestApp.elf tag microblaze_0 –o
   b download.bit
   ```

   Hint: Copy the command text from this document and paste it in the shell or command window by right-clicking and selecting **Paste**.

   This generates the `download.bit` in the `image\` directory.
4. In the Bash shell, execute the following command to generate the `system.ace` file in the `image\` directory.

   ```
   xmd -tcl genace.tcl -jprog -target mdm -hw download.bit -board ml605 -
   ace system.ace
   ```
5. Using Windows Explorer, copy and rename the following files, as shown in Table 1.

   Table 1: Renaming partial bit files

File Name	Copy to Directory	Rename File To
`reconfig_peripheral_lab\PlanAhead\PlanAhead.runs\adder\adder_math_0_math_0_user_logic_i_rp_instance_adder_partial.bit`	`\image`	`adder.bit`
`reconfig_peripheral_lab\PlanAhead\PlanAhead.runs\mult\mult_math_0_math_0_user_logic_i_rp_instance_mult_partial.bit`	`\image`	`mult.bit`
`reconfig_peripheral_lab\PlanAhead\PlanAhead.runs\black_box\black_box_math_0_math_0_user_logic_i_rp_instance_math_bb_partial.bit`	`\image`	`blank.bit`

Copying the system.ace and Three Partial Bit Files on a Compact Flash Memory Card

1. Place a blank Compact Flash memory card in a Compact Flash writer.
2. Using Windows Explorer, copy the three partial bit files and the `system.ace` file from `reconfig_peripheral_lab\image\` folder to the Compact Flash card.
3. Place the Compact Flash card in the ML605 board.
4. Set the SACE Mode pins (S1) to **0111 (dn-up-up-up)** to configure the FPGA device from the Compact Flash.
5. Connect your PC to the ML605 with the provided USB cable.
6. Install the driver, if necessary. For instructions, see the *ML605 Hardware User Guide*:

   http://www.xilinx.com/support/documentation/boards_and_kits/ug534.pdf
7. Start a HyperTerminal window, connecting using **COMx at 115200 baud** and power **ON** the ML605 board.
8. Press **CPU Reset**.
9. Follow the menu and test various reconfigurations.

Conclusion

In this tutorial, you created a processor system using XPS, added a user peripheral which included a place holder for the reconfigurable partition, and generated netlist files. Also, you created an application using SDK. Full bitstream as well as partial reconfiguration bitstreams were generated using the PlanAhead tool. Also, you generated an ACE file for Compact Flash memory card. You verified the functionality using the ML605 evaluation board.

Additional Resources

Xilinx Resources

- *ISE Design Suite: Installation and Licensing Guide* (UG798): http://www.xilinx.com/support/documentation/sw_manuals/xilinx13_4/iil.pdf
- *ISE Design Suite 13: Release Notes Guide* (UG631): http://www.xilinx.com/support/documentation/sw_manuals/xilinx13_4/irn.pdf
- **Product Support and Documentation**: http://www.xilinx.com/support/index.htm
- **Xilinx Glossary**: http://www.xilinx.com/company/terms.htm
- **Video Demonstrations**: http://www.xilinx.com/products/design_resources/design_tool/resources/index.htm

Partial Reconfiguration Documentation

- *Partial Reconfiguration User Guide* (UG702): http://www.xilinx.com/support/documentation/sw_manuals/xilinx13_4/ug702.pdf
- *Partial Reconfiguration Tutorial* (UG743): http://www.xilinx.com/support/documentation/sw_manuals/xilinx13_4/PlanAhead_Tutorial_Partial_Reconfiguration.pdf

PlanAhead Documentation

- *PlanAhead User Guide* (UG632): http://www.xilinx.com/support/documentation/sw_manuals/xilinx13_4/PlanAhead_UserGuide.pdf
- **PlanAhead Methodology Guides**: http://www.xilinx.com/support/documentation/dt_planahead_planahead13-4_userguides.htm
- **PlanAhead Tutorials**: http://www.xilinx.com/support/documentation/dt_planahead_planahead13-4_tutorials.htm