PlanAhead Software Tutorial

Partial Reconfiguration of a Processor Peripheral

UG744 (v 13.2) July 6, 2011

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.

Table of Contents

PlanAhead Software Tutorial: Partial Reconfiguration of a Processor Peripheral

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™ software. 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 XPS 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 software 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 XPS HWICAP peripheral.

Getting Started

Software Requirements

The PlanAhead software is installed with the ISE Design Suite 13.2 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/xilinx13.2/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_planahead13-2_tutorials.htm

Understanding the Processor System

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

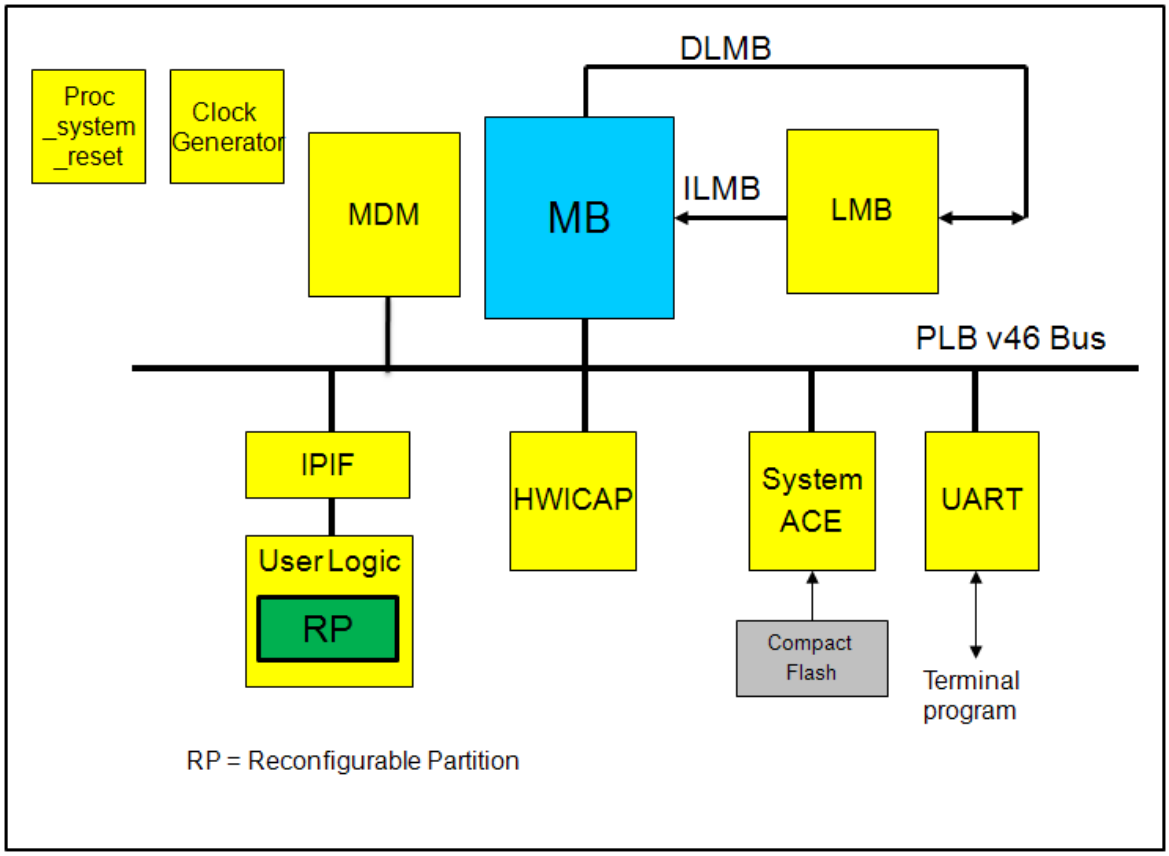

Figure 1: Top-Level Design

Project Directory Structure

The directory structure is:

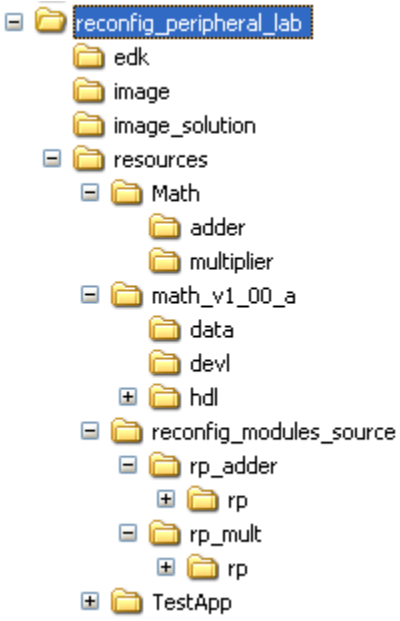

Figure 2: The Project Directory

- The `edk\` directory is used to create a processor system.
- The `resources\` directory contains:
  - Source files used to generate the netlists of the addition and multiplication functions,
  - A pre-compiled netlist for the addition and multiplication functions in Math and associate sub-directories, and
  - A software application to demonstrate the functionality.
- The math processor core (pcore) that:
  - Provides necessary processor bus connections
  - Provides the required peripheral services (in this case, one slave register and a software reset)
  - 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 13.2 > EDK > Xilinx Platform Studio** to open XPS.
2. In the dialog box that opens, select the option to create a new project using the Base System Builder wizard and click **OK**.
3. Browse to the `reconfig_peripheral_lab\edk\` directory.
4. Click **Save**.

   A dialog box will appear indicating the type of interconnect the system will have.
5. Select **PLB System** and click **OK**.
6. Click **Next** to create a new processor system.

   You will create a system for a Virtex®-6 ML605 evaluation platform.
7. In Board Vendor, select **Xilinx**.
8. In Board Name, select **Virtex 6 ML605 Evaluation Platform**.
9. In Board Revision, select **D**.
10. Click **Next**.
11. Select **a Single-Processor System**.
12. Click **Next**.
13. Select **83.33 MHz** from the System Clock Frequency drop-down menu.
14. Select **64 KB** from the Local Memory drop-down menu.
15. Click **Next**.
16. In the selected peripherals list on the right, remove all devices *except*:
    - RS232_Uart_1
    - SysACE_CompactFlash
    - dlmb_cntlr
    - ilmb_cntlr
17. Click **RS232_Uart_1** and configure it with a baud rate of **115200**.
18. Click **Next**.
19. Click **Next** as we do not have any cacheable memory.
20. In the summary screen, click **Finish**.
21. 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.

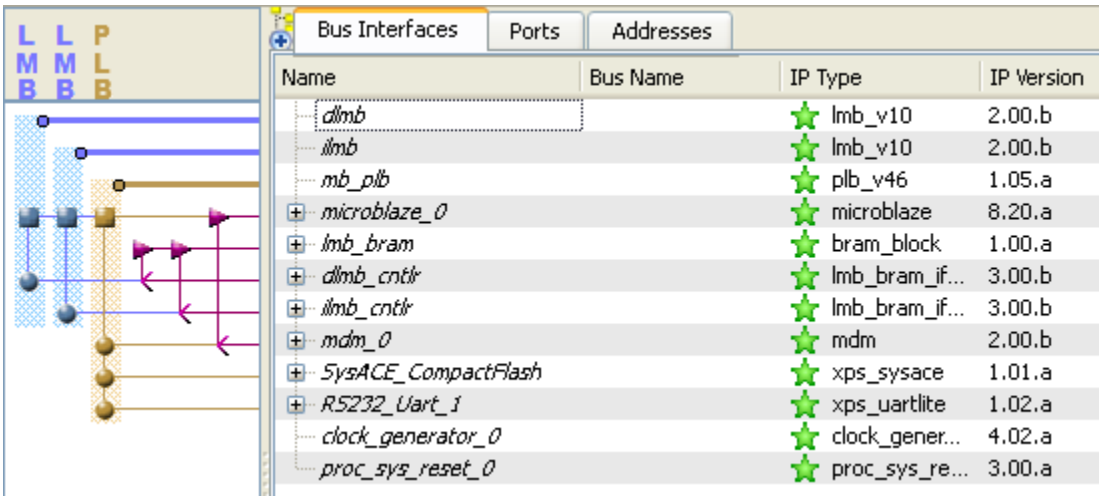

Figure 3: Displaying the System Assembly View

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 131. The same is instantiated at line 156. The data inputs to the component are clocked at lines starting at 189. 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 395 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. Drag MATH to the System Assembly View.
6. A properties form will be displayed. Click **OK** to accept the default settings.

7. In the IP Catalog tab, select the `FPGA Internal Configuration Access Port (v5.01.a)` IP 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** to accept the default settings.

Connecting the Buses, and Generating the Addresses

1. In the System Assembly View, expand the `math_0` and `xps_hwicap_0` instances.
2. Connect the various buses shown in Figure 4.

   To connect a peripheral to a bus, move the mouse in the bus window and click a corresponding unfilled circle or square. A peripheral is connected when the circle or rectangle is filled.

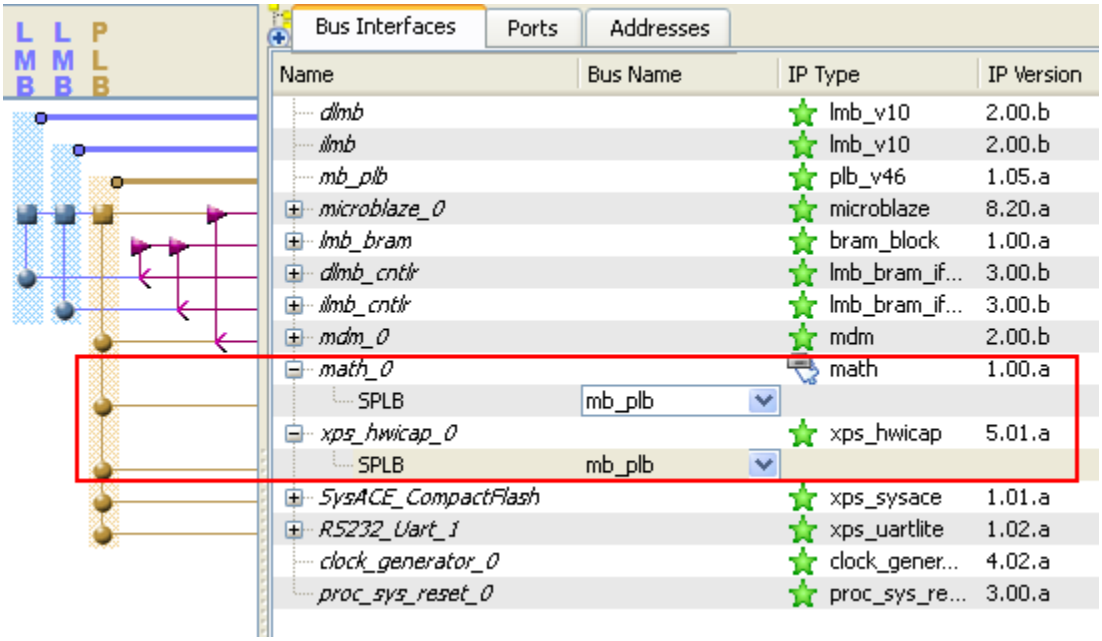

Figure 4: Displaying the Bus Connections in System Assembly View

3. From the Addresses tab, click **Generate Addresses** to assign the addresses to the `math_0` and `xps_hwicap_0` instances.

   The resulting address map should appear as shown in the following figure.

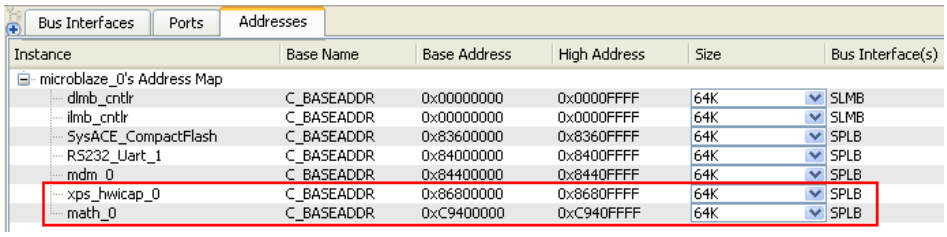

Figure 5: Verifying the Address Map of the System Peripherals

Connecting the Ports

1. In the System Assembly View, select the **Ports** tab.
2. Expand the `xps_hwicap_0` instance.
3. Click the drop-down button of the **ICAP_Clk**.
4. Select **clk_83_3333MHz**.

   The connection appears as shown in Figure 6.

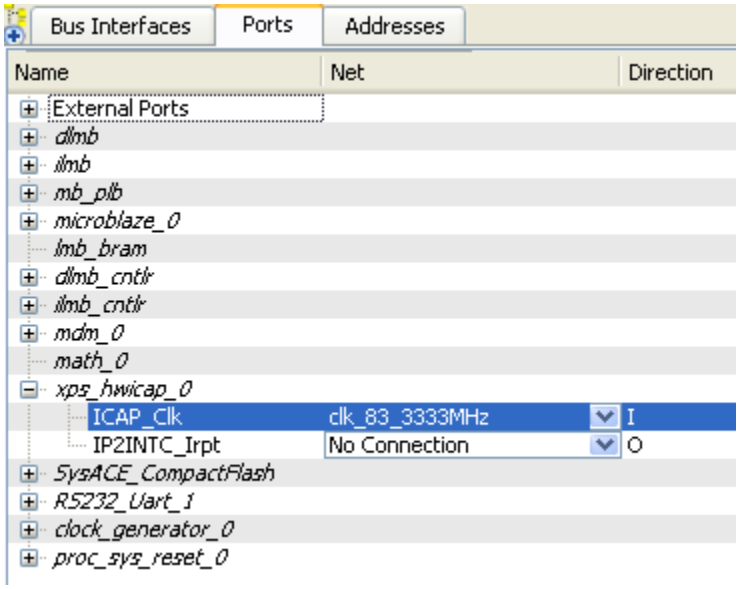

Figure 6: Connecting Clock Source to ICAP

Partial Reconfiguration Design Details

The `xps_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 83.33 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 software.

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**.
4. A workspace location dialog box will appear.
5. Browse to the `reconfig_peripheral_lab\edk\` directory and click **OK**.
6. Click **OK** to create SDK directory and open SDK.
7. In SDK, select **File > New > Xilinx Board Support Package**.
8. Notice that the default Project Name is `Standalone_bsp_0` and the OS is `Standalone`.
9. Click **Finish** with default settings.

   The Board Support Package Settings window opens.

11. Check the xilfatfs check box to select the FAT file system support for the Compact Flash card.

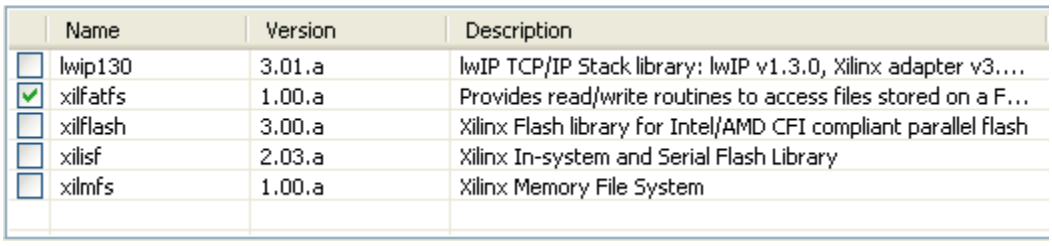

Figure 7: Selecting File System Support

12. 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**.

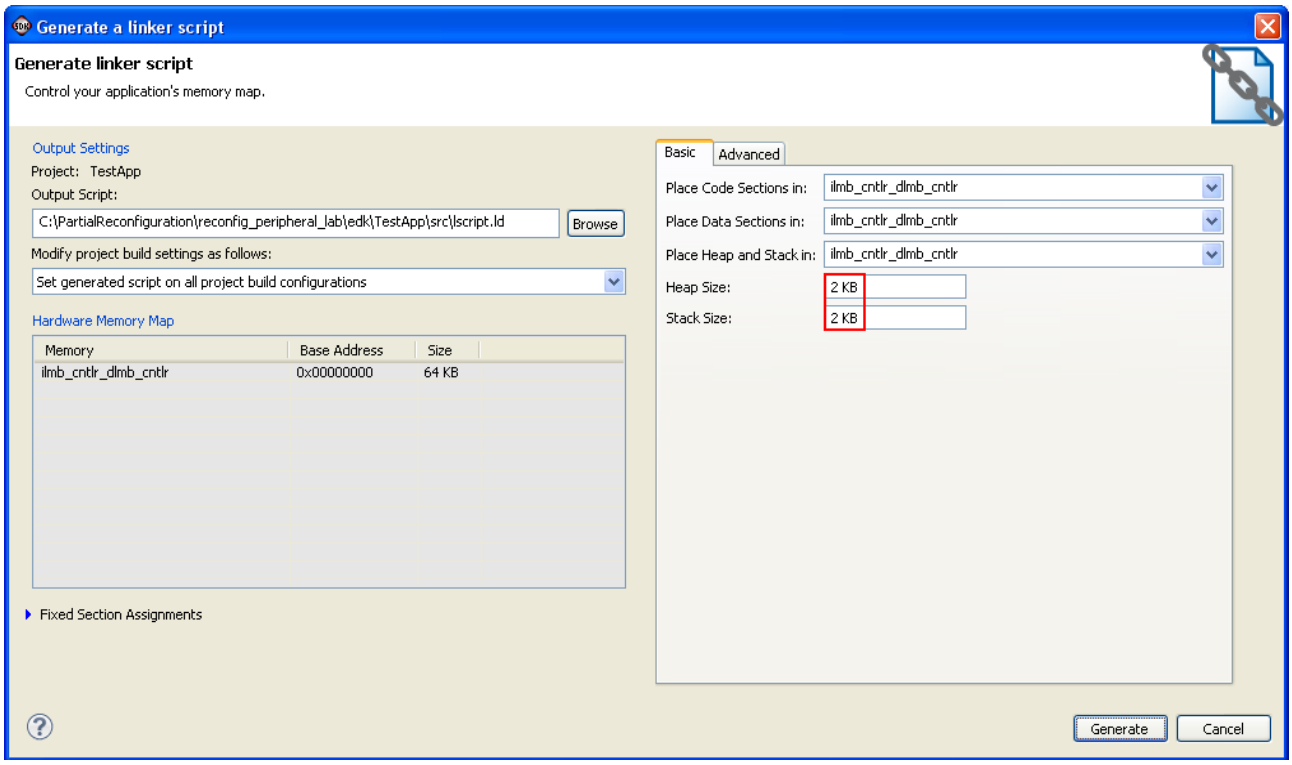

Figure 8: Generating a Linker Script

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 PlanAhead 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 13.2 > 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.

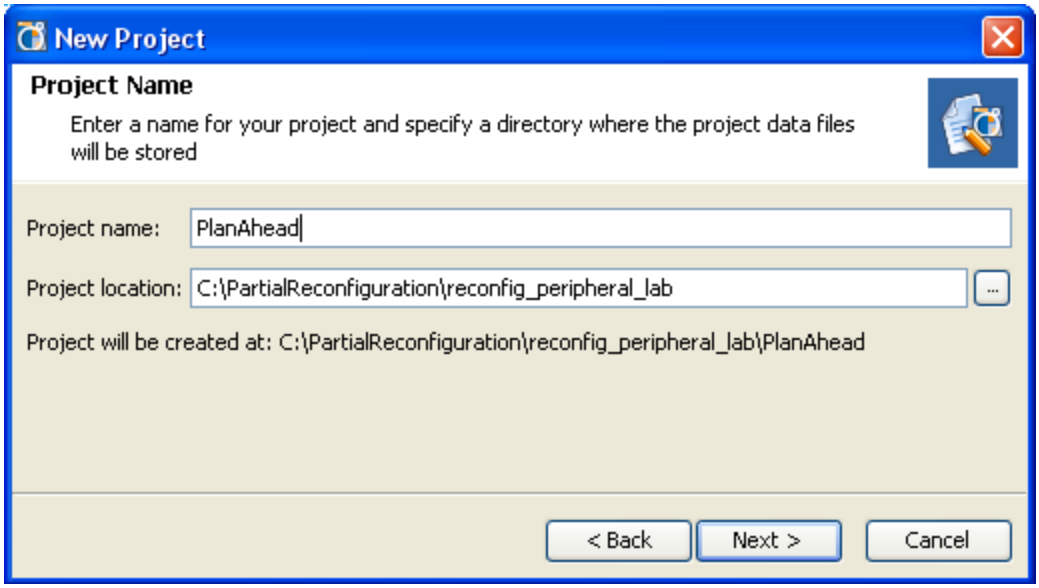

Figure 9: Project Name Page of the New Project Wizard

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

9. Check the **Set PR Project** option.

   Note: If you forget to check the option, you can still enable it from the project by selecting **Tools > Project Options** and clicking the check box. This must be done before a partition can be defined as reconfigurable.

10. Click **Next**.

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

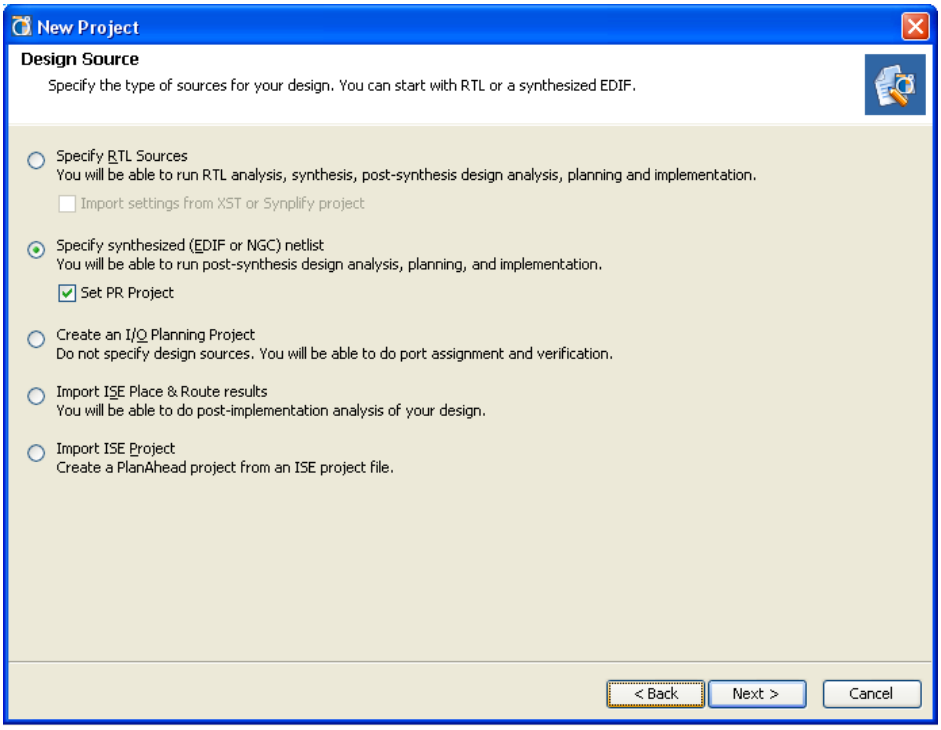

Figure 10: Importing Synthesized Netlists

11. Browse to `reconfig_peripheral_lab\edk\implementation`.
12. Select the `system.ngc` file and click **Open**.

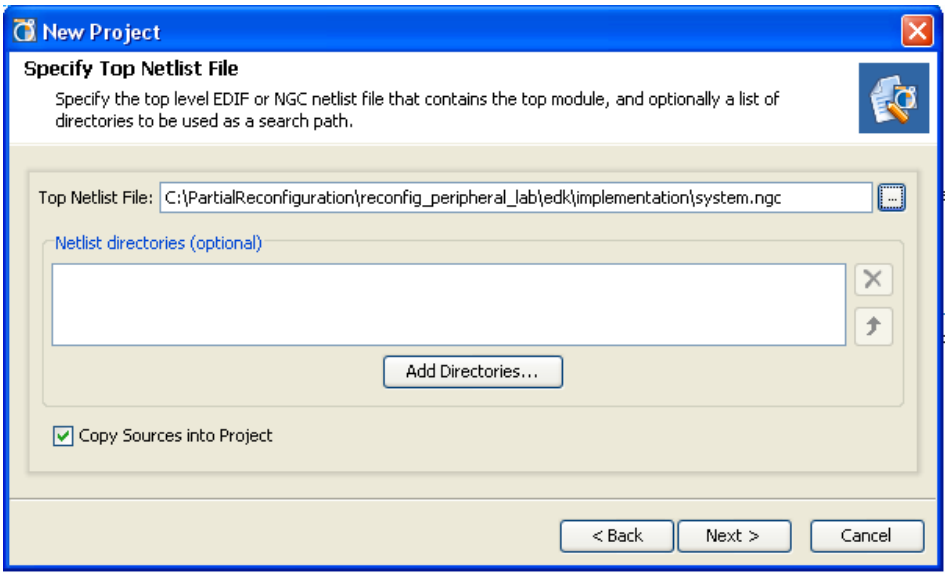

Figure 11: Selecting the Top-level Netlist File

13. Click **Next**.

    The Add Constraint files (optional) page opens.

14. Select **UCF**.
15. Click **Add Files**.
16. Browse to `reconfig_peripheral_lab\edk\data\`
17. Select **system.ucf**.
18. Click **Open**.
19. Click **Next** to open the Product Family and Default Part page.
20. 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.

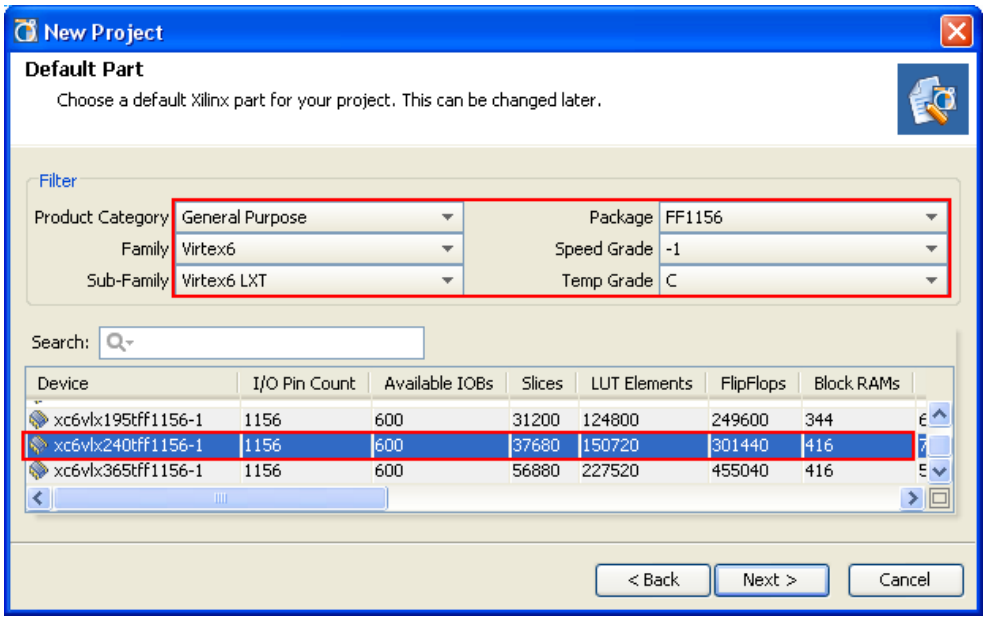

Figure 12: Selecting the Target Device

21. Click **Next**.
22. Click **Finish**.

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

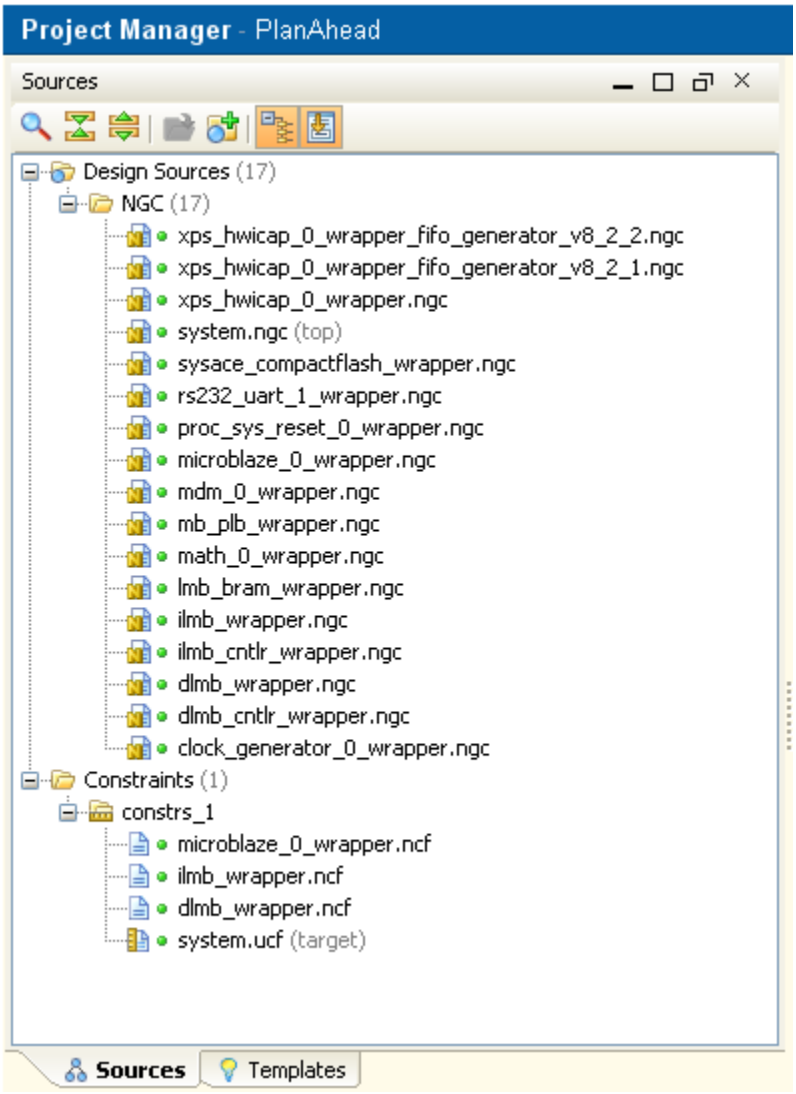

Figure 13: Design Hierarchy in PlanAhead

Step 4: Defining a Reconfigurable Partition

This design has one reconfigurable partition that must be explicitly 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, as the netlist file for it is missing. This is expected, as no netlist has been associated with this module yet.

   A Netlist tab displays the hierarchical view of the system, as shown in Figure 14.

2. Click **OK**.
3. Expand the `math_0` instance.
4. Select **math_0/USER_LOGIC_I/rp_instance** in the Netlist tab.

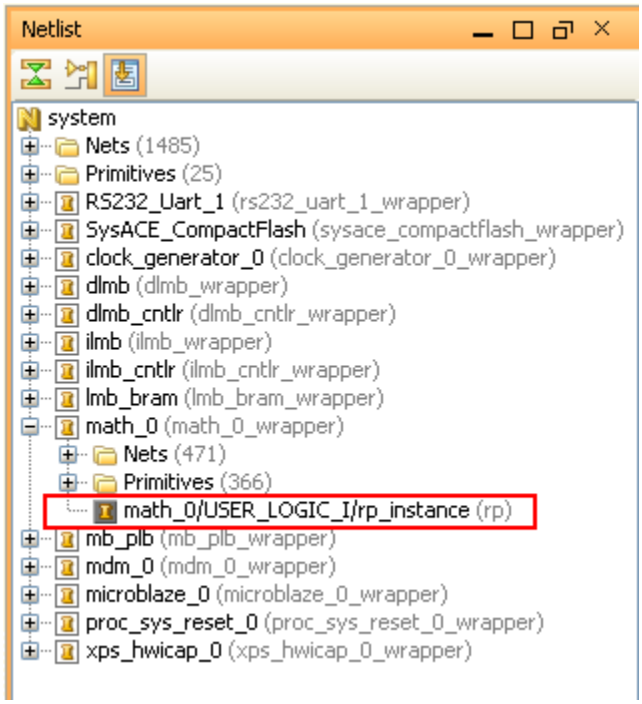

Figure 14: Netlists Hierarchy View

5. Right-click and select **Set Partition**.
6. Click **Next** *twice*.

   The Set Partition dialog box will appear, as shown in Figure 15.

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.

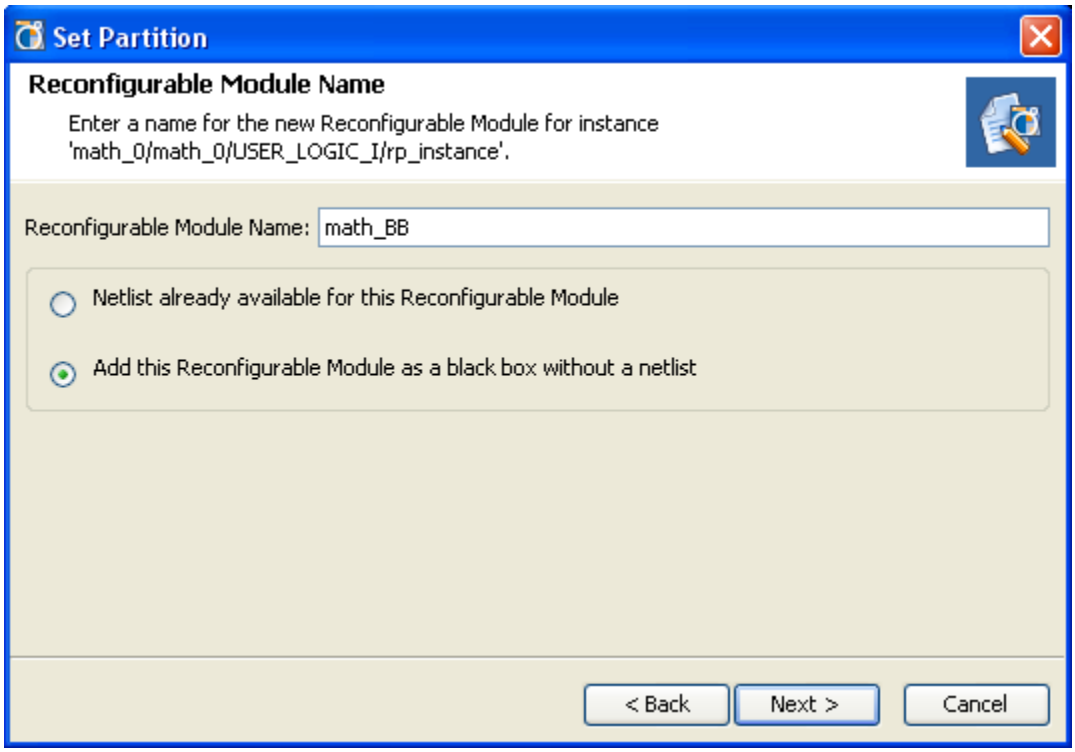

Figure 15: Setting a Partition

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, as shown in Figure 16.
4. In the Reconfigurable Module Name field, type **`adder`**.
5. Verify that **Netlist already available for this Reconfigurable Module** is selected.

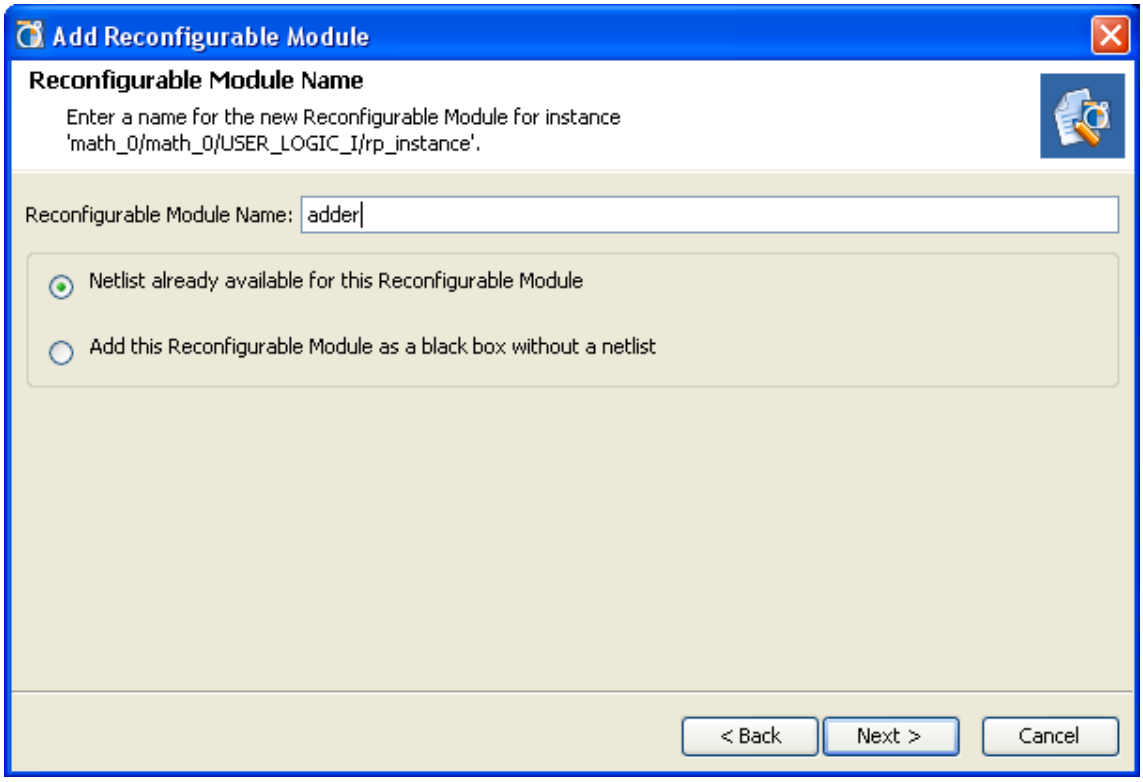

Figure 16: Adding a Reconfigurable Module

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

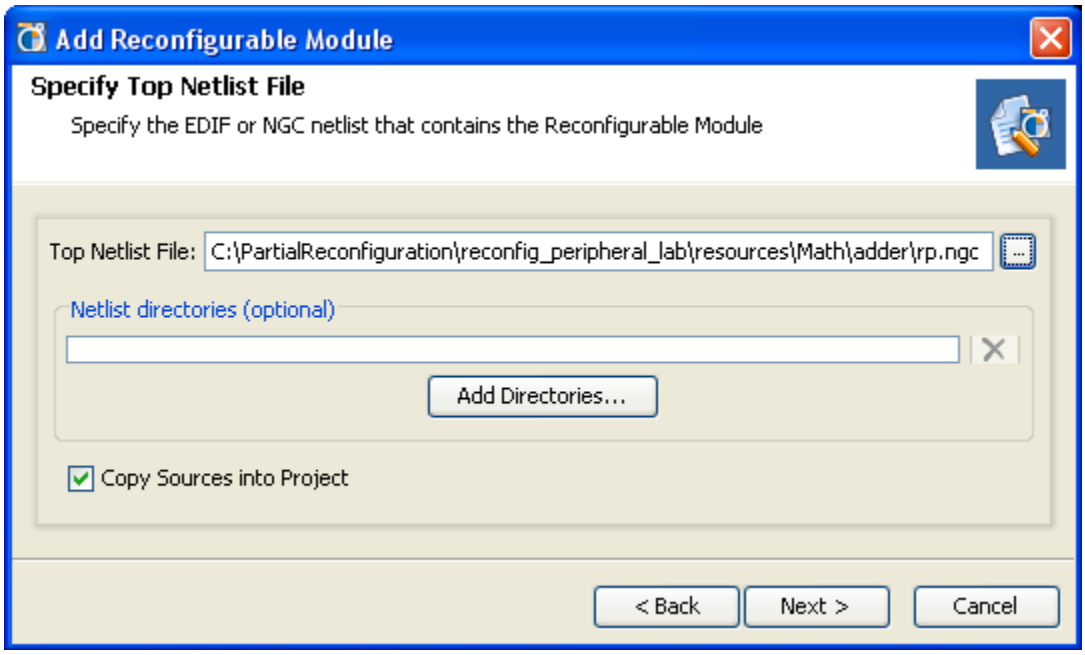

Figure 17: Locating the adder Version of math.ngc

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.

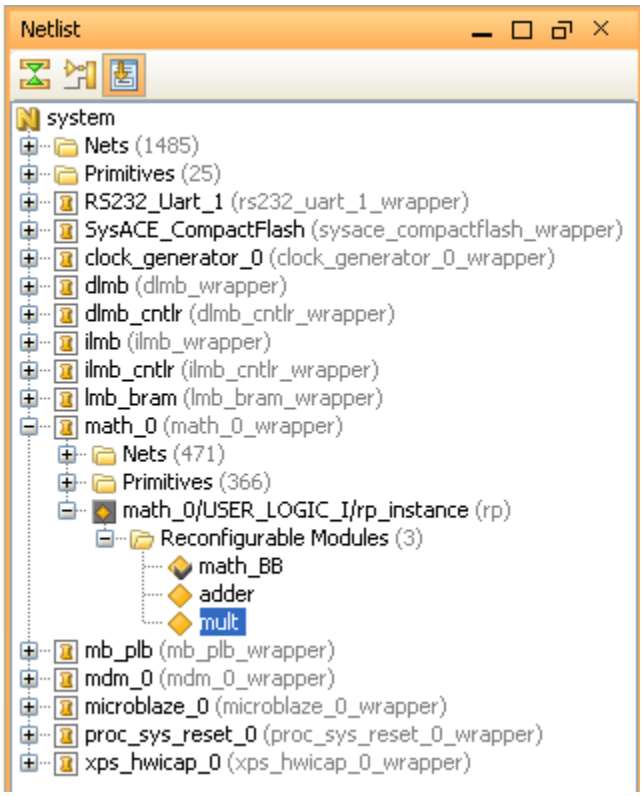

Figure 18: PlanAhead Project with adder and mult RMs Added

Step 6: Defining the Reconfigurable Partition Region

You must now 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**.

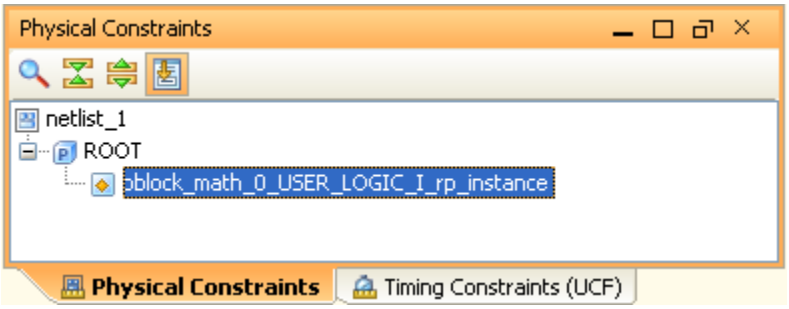

Figure 19: Setting Physical Constraints

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.

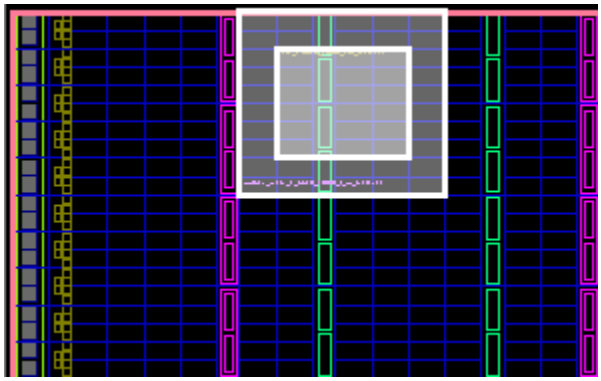

Figure 20: Closer View of the Pblock Area

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**.

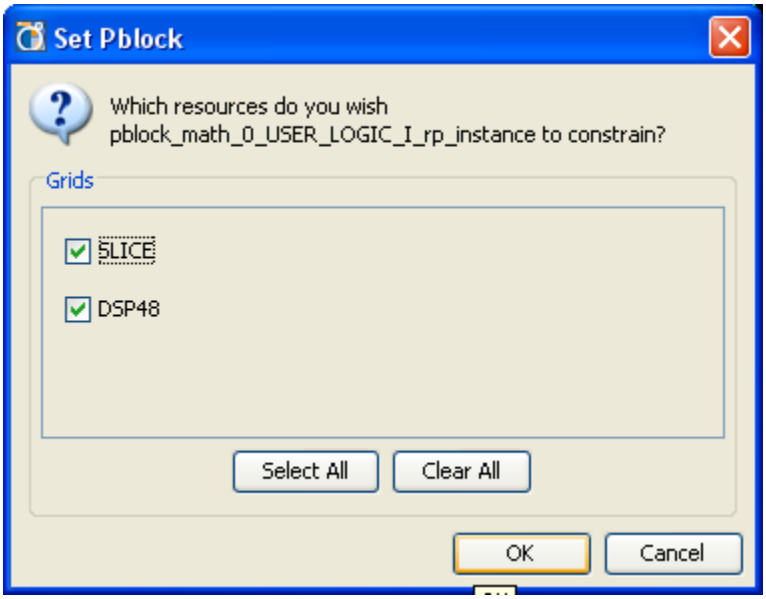

Figure 21: Setting Pblock with SLICE and DSP48

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.

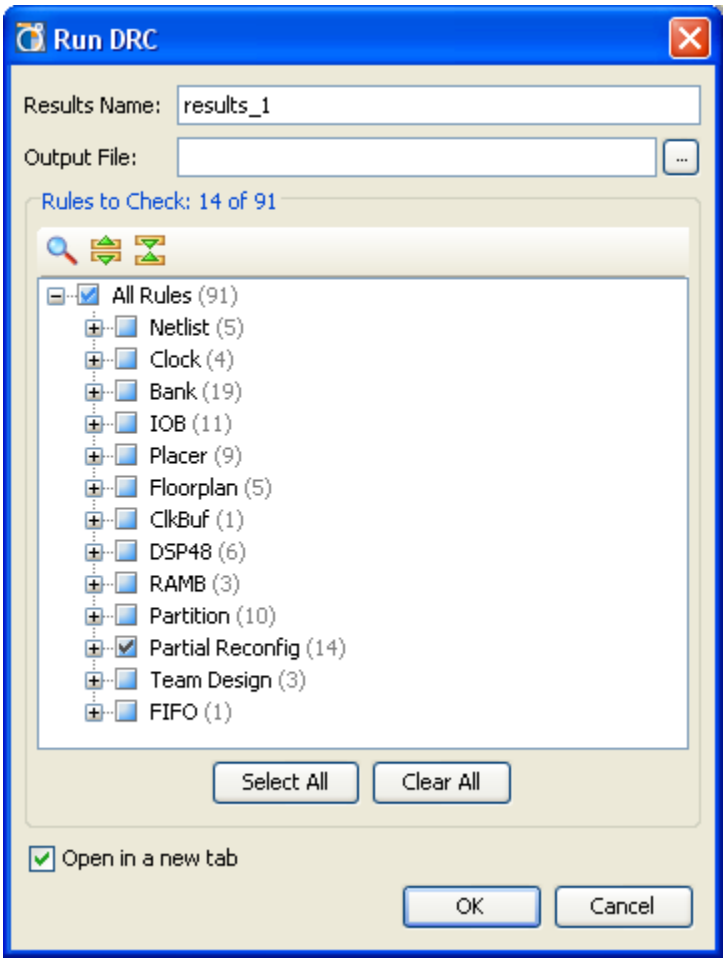

Figure 22: Running Design Rule Checks

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.

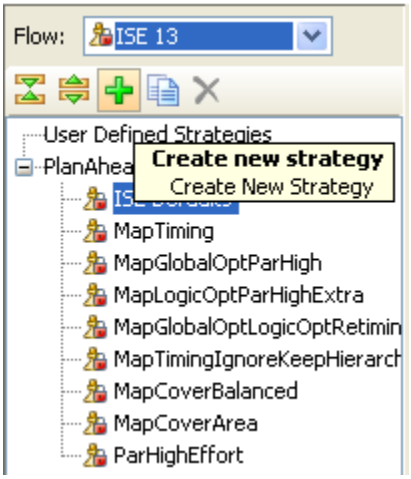

Figure 23: Creating a New Strategy

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 GUI, 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`.

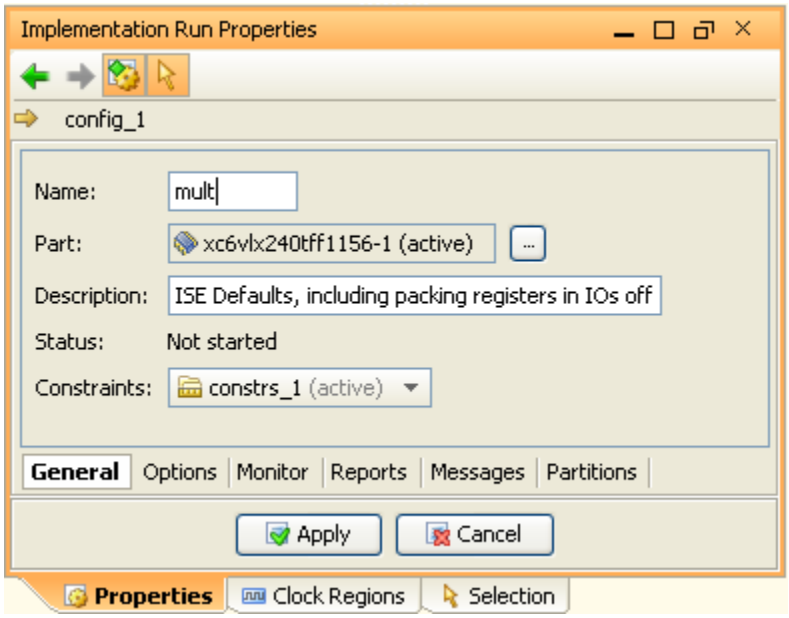

Figure 24: Implementation Run Properties View

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, as shown in Figure 25.
9. Click **Apply**.

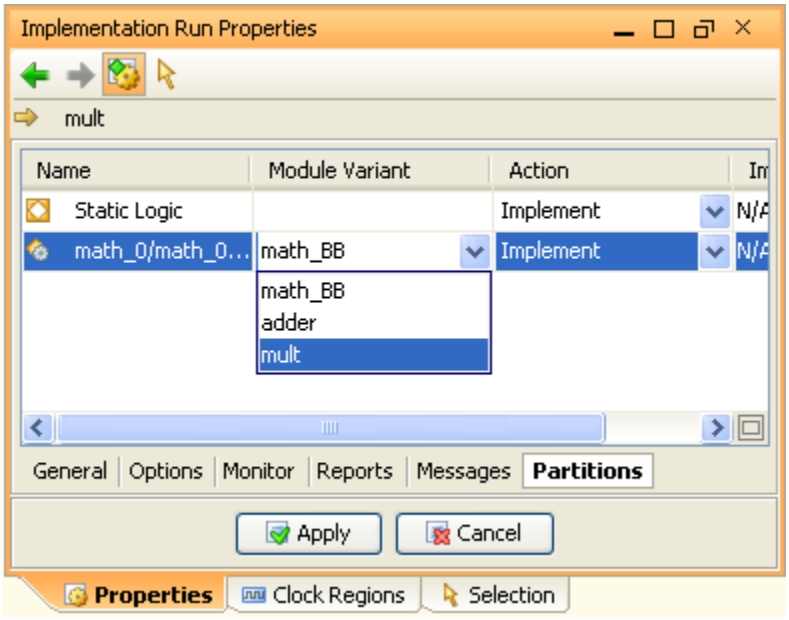

Figure 25: Implementation Run

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 that enables you to 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.

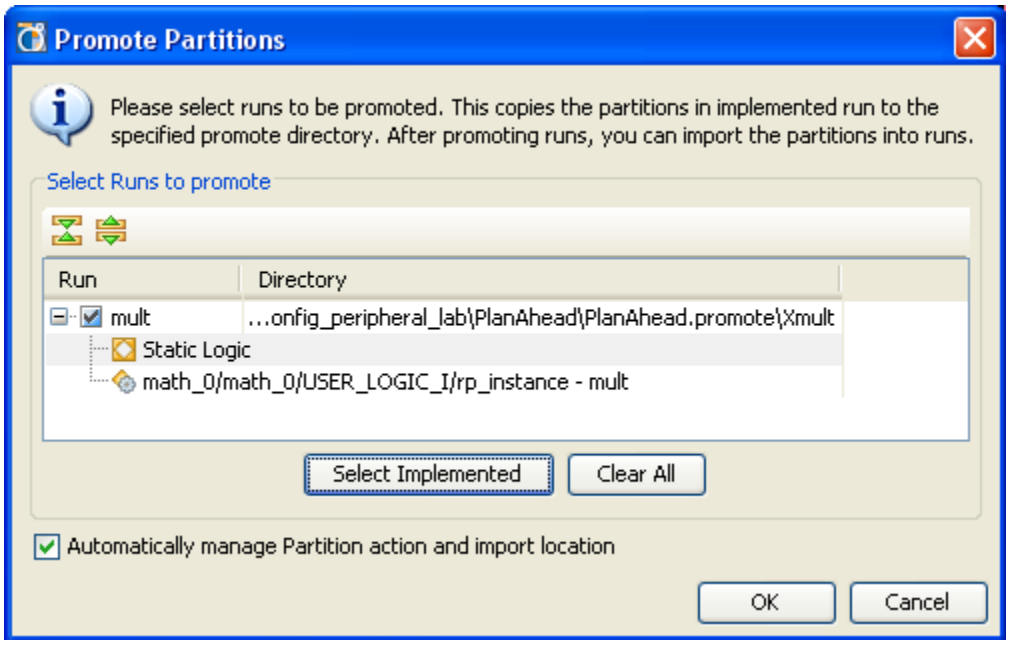

Figure 26: Promoting Partitions

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. Close the project, and re-open the project.

   This step is required for 13.2 version in order for the tools to see that the implemented mult configuration is promoted. Ignoring this step will generate error when new configuration runs are created and tried to run implementation.

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

   The Create Multiple Runs window opens.

3. Click **Next** *twice*.

4. In the Choose Implementation Strategies and Reconfigurable Modules page, change the name of the configuration from `config_1` to **`adder`**.

5. Click **More**.

6. Change the name of `config_1` to **`black_box`**.

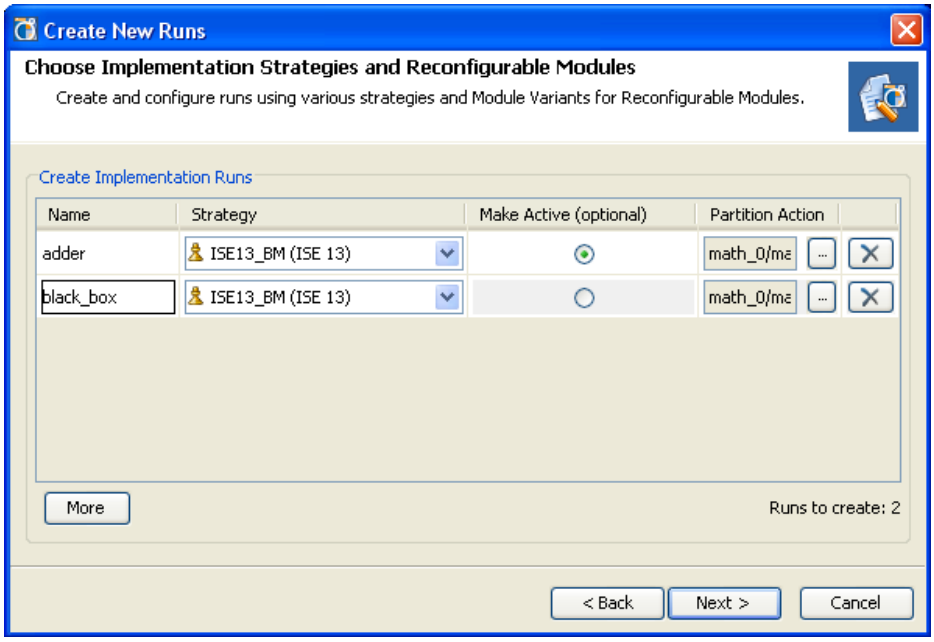

Figure 27: Creating Multiple Runs

7. In the adder configuration row, click the **Partition Action** field.

8. For the rp_instance row, click the Module Variant column drop-down arrow, and select **adder** as the variant to be implemented (Figure 28).

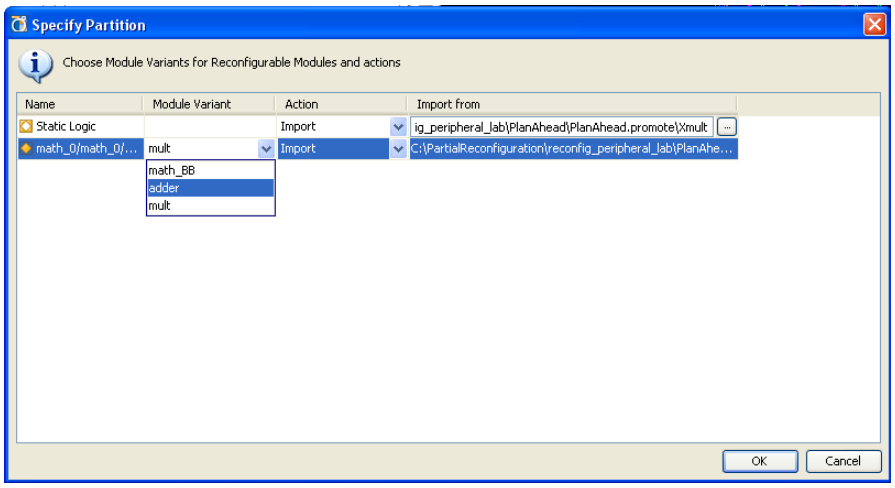

Figure 28: Selecting adder Module Variant

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

Step 10: Running Partial Reconfiguration to Verify Utility

You must 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**.

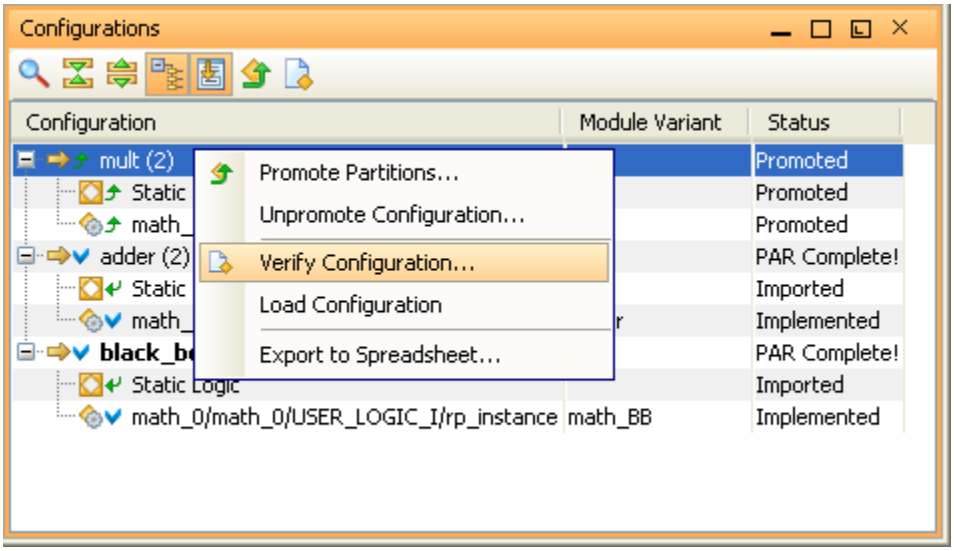

Figure 29: Verifying All Configurations

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 Bash Shell**, or
   - From your Windows environment, select **Start > Programs > Xilinx ISE Design Suite 13.2 > Accessories > ISE Design Suite Command Prompt**.
2. In the Bash 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/TestApp/Debug/TestApp.elf tag microblaze_0 -o b download.bit
   ```

   Hint: Copy the command text from this document and paste it in the Bash shell by right-clicking on the title bar and selecting **Edit > 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 software. Also, you generated an ACE file for Compact Flash memory card. You verified the functionality using the ML605 evaluation board.

Appendix A

Additional Resources

Xilinx Resources

- ***ISE Design Suite: Installation and Licensing Guide*** *(UG798):*
  http://www.xilinx.com/support/documentation/sw_manuals/xilinx13_2/iil.pdf
- ***ISE Design Suite 13: Release Notes Guide*** *(UG631):*
  http://www.xilinx.com/support/documentation/sw_manuals/xilinx13_2/irn.pdf
- **Xilinx® Documentation**:
  http://www.xilinx.com/support/documentation.htm
- **Xilinx Glossary**:
  http://www.xilinx.com/support/documentation/sw_manuals/glossary.pdf
- **Xilinx Support**:
  http://www.xilinx.com/support
- **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_2/ug702.pdf
- ***PlanAhead Software Tutorial: Overview of the Partial Reconfiguration Flow*** *(UG743):*
  http://www.xilinx.com/support/documentation/sw_manuals/xilinx13_2/PlanAhead_Tutorial_Partial_Reconfiguration.pdf

PlanAhead Documentation

- ***PlanAhead User Guide*** *(UG632):*
  http://www.xilinx.com/support/documentation/sw_manuals/xilinx13_2/PlanAhead_UserGuide.pdf
- **PlanAhead Methodology Guides**:
  http://www.xilinx.com/support/documentation/dt_planahead_planahead13-2_userguides.htm
- **PlanAhead Tutorials**:
  http://www.xilinx.com/support/documentation/dt_planahead_planahead13-2_tutorials.htm