

PlanAhead Software Tutorial

Partial Reconfiguration of a Processor Peripheral

UG 744 (v 12.2) July 23, 2010





Xilinx is disclosing this Document and Intellectual Property (hereinafter "the Design") to you for use in the development of designs to operate on, or interface with Xilinx FPGAs. Except as stated herein, none of the Design may be copied, reproduced, distributed, republished, downloaded, displayed, posted, or transmitted in any form or by any means including, but not limited to, electronic, mechanical, photocopying, recording, or otherwise, without the prior written consent of Xilinx. Any unauthorized use of the Design may violate copyright laws, trademark laws, the laws of privacy and publicity, and communications regulations and statutes.

Xilinx does not assume any liability arising out of the application or use of the Design; nor does Xilinx convey any license under its patents, copyrights, or any rights of others. You are responsible for obtaining any rights you may require for your use or implementation of the Design. Xilinx reserves the right to make changes, at any time, to the Design as deemed desirable in the sole discretion of Xilinx. Xilinx assumes no obligation to correct any errors contained herein or to advise you of any correction if such be made. Xilinx will not assume any liability for the accuracy or correctness of any engineering or technical support or assistance provided to you in connection with the Design.

THE DESIGN IS PROVIDED "AS IS" WITH ALL FAULTS, AND THE ENTIRE RISK AS TO ITS FUNCTION AND IMPLEMENTATION IS WITH YOU. YOU ACKNOWLEDGE AND AGREE THAT YOU HAVE NOT RELIED ON ANY ORAL OR WRITTEN INFORMATION OR ADVICE, WHETHER GIVEN BY XILINX, OR ITS AGENTS OR EMPLOYEES. XILINX MAKES NO OTHER WARRANTIES, WHETHER EXPRESS, IMPLIED, OR STATUTORY, REGARDING THE DESIGN, INCLUDING ANY WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE, AND NONINFRINGEMENT OF THIRD-PARTY RIGHTS.

IN NO EVENT WILL XILINX BE LIABLE FOR ANY CONSEQUENTIAL, INDIRECT, EXEMPLARY, SPECIAL, OR INCIDENTAL DAMAGES, INCLUDING ANY LOST DATA AND LOST PROFITS, ARISING FROM OR RELATING TO YOUR USE OF THE DESIGN, EVEN IF YOU HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. THE TOTAL CUMULATIVE LIABILITY OF XILINX IN CONNECTION WITH YOUR USE OF THE DESIGN, WHETHER IN CONTRACT OR TORT OR OTHERWISE, WILL IN NO EVENT EXCEED THE AMOUNT OF FEES PAID BY YOU TO XILINX HEREUNDER FOR USE OF THE DESIGN. YOU ACKNOWLEDGE THAT THE FEES, IF ANY, REFLECT THE ALLOCATION OF RISK SET FORTH IN THIS AGREEMENT AND THAT XILINX WOULD NOT MAKE AVAILABLE THE DESIGN TO YOU WITHOUT THESE LIMITATIONS OF LIABILITY.

The Design is not designed or intended for use in the development of on-line control equipment in hazardous environments requiring fail-safe controls, such as in the operation of nuclear facilities, aircraft navigation or communications systems, air traffic control, life support, or weapons systems ("High-Risk Applications") Xilinx specifically disclaims any express or implied warranties of fitness for such High-Risk Applications. You represent that use of the Design in such High-Risk Applications is fully at your risk.

© 2010 Xilinx, Inc. All rights reserved. XILINX, the Xilinx logo, and other designated brands included herein are trademarks of Xilinx, Inc. All other trademarks are the property of their respective owners.

Demo Design License

© 2010 Xilinx, Inc.

This Design is free software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation; either version 2.1 of the License, or (at your option) any later version.

This library is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.

You should have received a copy of the GNU Library General Public License along with this design file; if not, see: <http://www.gnu.org/licenses/>



The PlanAhead™ software source code includes the 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 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 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.

Copyright (C) 2000-2003, ASICs World Services, LTD. AUTHORS

All rights reserved.

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

Partial Reconfiguration of a Processor Peripheral	7
Introduction	7
Sample Design Data	7
Xilinx ISE Design Suite and PlanAhead Software	7
Required Hardware	8
PlanAhead Documentation and Information	8
Tutorial Objectives	8
Tutorial Procedure	8
Directory Structure	9
Tutorial Steps	10
Step 1: Create a Processor Hardware System Step 1	11
Step 2: Create a Software Project Step 2	15
Step 3: Create a PlanAhead Project Step 3	18
Step 4: Define a Reconfigurable Partition Step 4	22
Step 5: Add Reconfigurable Modules Step 5	24
Step 6: Define the Reconfigurable Partition Region Step 6	27
Step 7: Run Design Rule Checker Step 7	30
Step 8: Create First Configuration, Implement, and Promote Step 8	31
Step 9: Create Other Configurations and Implement Step 9	34
Step 10: Run Partial Reconfiguration to Verify Utility Step 10	36
Step 11: Generate Bit Files Step 11	37
Step 12: Create Image and Test Step 12	38
Conclusion	39

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) and the PlanAhead™ software. You will use XPS to create a processor system which includes a lower-level module defining one Reconfigurable Partition (RP) and two Reconfigurable Modules (RM). The two RM perform addition and multiplication functions.

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

You will use PlanAhead to:

1. Floorplan the design including defining a reconfigurable partition for the reconfigurable region
2. 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 has been validated with ISE Design Suite Release 12.2. Ensure that your installation is set to this revision or newer before beginning.

This tutorial covers only a subset of the features contained in the PlanAhead software bundled with ISE Design Suite Release 12.2. Other features are covered in other tutorials.

If you have any questions or issues with this tutorial, contact Xilinx Technical Support.

Sample Design Data

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 <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, or go to the Xilinx website at: <http://www.xilinx.com/getproduct>

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

Xilinx ISE Design Suite and PlanAhead Software

The PlanAhead software is installed with the ISE Design Suite 12.2 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 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 a minimum of 2 GB of RAM for use with this design to obtain a reasonable performance.

PlanAhead Documentation and Information

For information about the PlanAhead software, please see the following documents:

- *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 information about Partial Reconfiguration, refer to the Partial Reconfiguration documentation, including the *Partial Reconfiguration User Guide* (UG702), at: <http://www.xilinx.com/tools/partial-reconfiguration>

Tutorial Objectives

After completing this tutorial, you will be able to:

- Generate a processor system using XPS.
- 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.

Tutorial Procedure

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.

The user verifies the functionality with HyperTerminal under user application control. The dynamic modules are reconfigured using the XPS HWICAP peripheral.

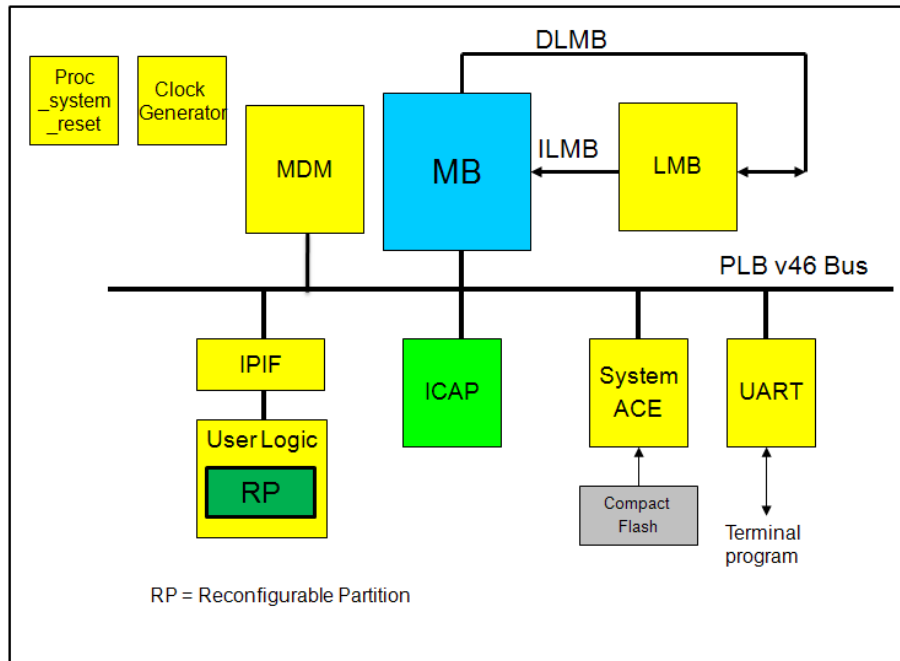


Figure 1: Top-Level Design

Directory Structure

The directory structure is:

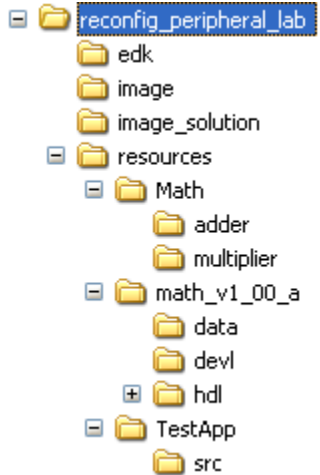


Figure 2: The Project Directory

- The `\edk` directory is used to create a processor system.
- The `\resources` directory contains:
 - 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 No.	Step Name
----------	-----------

- | | |
|----|--|
| 1 | Create a Processor Hardware System |
| 2 | Create a Software Project |
| 3 | Create a PlanAhead Project |
| 4 | Define a Reconfigurable Partition |
| 5 | Add Reconfigurable Modules |
| 6 | Define the Reconfigurable Partition Region |
| 7 | Run Design Rule Checker |
| 8 | Create First Configuration and Implement |
| 9 | Create Other Configurations and Implement |
| 10 | Run Partial Reconfiguration Utility |
| 11 | Generate Bit Files |
| 12 | Create an Image and Test |

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: Create a Processor Hardware System

Step 1

1-1. Create a processor system using the Base System Builder (BSB) wizard in XPS.

1-1-1. Select **Start > Programs > Xilinx Design Suite 12.2 > EDK > Xilinx Platform Studio** to open XPS.

1-1-2. In the dialog box that opens, select the option to create a new project using the Base System Builder wizard and click **OK**.

1-1-3. Browse to the `reconfig_peripheral_lab\edk` directory.

1-1-4. Click **Save**.

1-1-5. Click **OK**.

1-1-6. Click **Next** to create a new processor system.

You will create a system for a Virtex[®]-6 ML605 evaluation platform.

1-1-7. In Board Vendor, select **Xilinx**.

1-1-8. In Board Name, select **Virtex 6 ML605 Evaluation Platform**.

1-1-9. In Board Revision, select **D**.

1-1-10. Click **Next**.

1-1-11. Select a Single-Processor System.

1-1-12. Click **Next**.

1-1-13. Set System Clock Frequency to **83.33 MHz**.

1-1-14. Select **64 KB** from the Local Memory drop-down menu.

1-1-15. Click **Next**.

1-1-16. In the selected peripherals list on the right, remove all devices *except*:

- RS232_Uart_1
- SysACE_CompactFlash
- dlmb_cntlr
- ilmb_cntlr

1-1-17. Click **RS232_Uart_1** and configure it with a baud rate of **115200**.

1-1-18. Click **Next**.

1-1-19. Review the next two screens, and click **Next** on each.

1-1-20. In the summary screen, click **Finish**.

1-1-21. On The Next Step dialog box, click **OK** to start using Platform Studio and open the System Assembly View window as shown in the following figure.

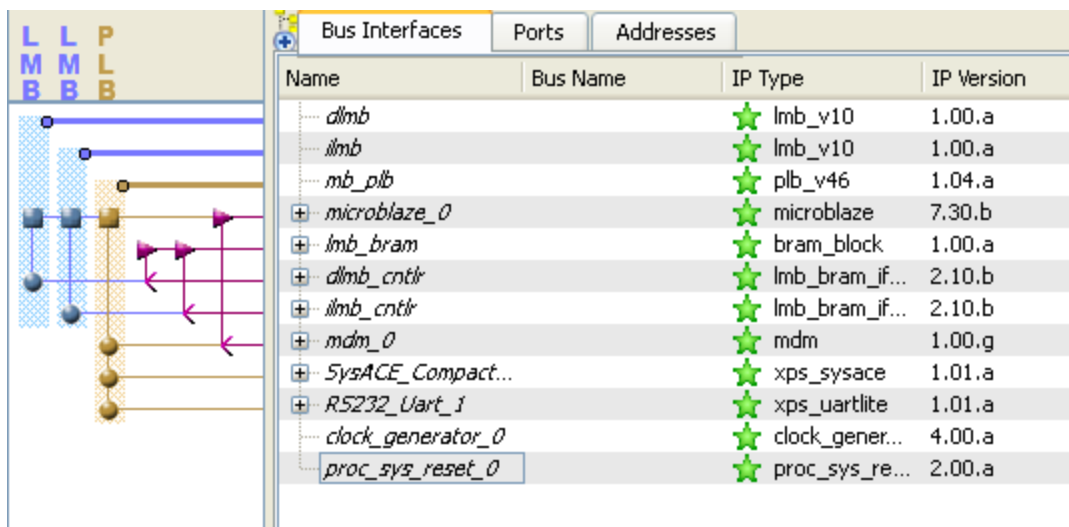


Figure 3: Displaying the System Assembly View

1-2. Add required IPs to the processor system.

- 1-2-1. 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

`reconfig_peripheral_lab\resources\math_v1_00_a\hdl\vhdl\user_logic.vhd` file. 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**.

- 1-2-2. 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.

- 1-2-3. Expand the `USER` folder.

- 1-2-4. Select **MATH**.

- 1-2-5. Drag MATH to the System Assembly View.

- 1-2-6. A properties form will be displayed. Click **OK** to accept the default settings.

- 1-2-7. In the IP Catalog tab, select the `FPGA Internal Configuration Access Port (v4.00.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.

- 1-2-8. Click **OK** to accept the default settings.

1-3. Connect the busses and generate the addresses.

1-3-1. In the System Assembly View, expand the `math_0` and `xps_hwicap_0` instances.

1-3-2. Connect the various busses 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.

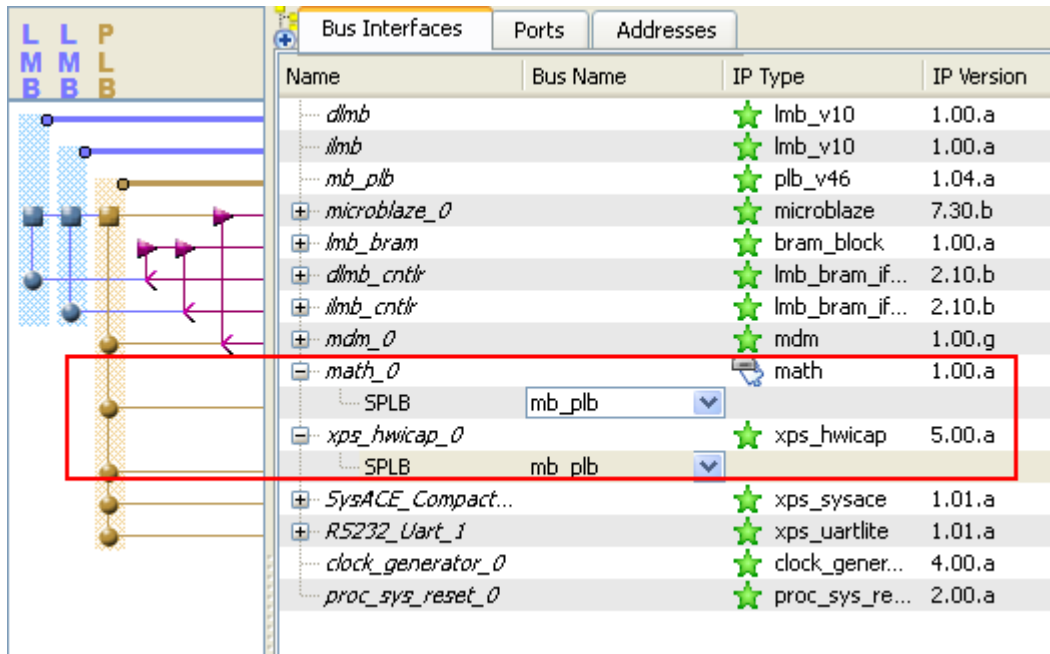


Figure 4: Displaying the Bus Connections in System Assembly View

1-3-3. 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.

Bus Interfaces Ports Addresses					
Instance	Base Name	Base Address	High Address	Size	Bus Int
microblaze_0's Address Map					
dlmb_cntlr	C_BASEADDR	0x00000000	0x0000FFFF	64K	SPLB
ilmb_cntlr	C_BASEADDR	0x00000000	0x0000FFFF	64K	SPLB
SysACE_CompactFlash	C_BASEADDR	0x83600000	0x8360FFFF	64K	SPLB
RS232_Uart_1	C_BASEADDR	0x84000000	0x8400FFFF	64K	SPLB
mdm_0	C_BASEADDR	0x84400000	0x8440FFFF	64K	SPLB
xps_hwicap_0	C_BASEADDR	0x86800000	0x8680FFFF	64K	SPLB
math_0	C_BASEADDR	0xC9400000	0xC940FFFF	64K	SPLB

Figure 5: Verifying the Address Map of the System Peripherals

1-4. Connect the ports.

1-4-1. In the **System Assembly View**, select the **Ports** tab.

1-4-2. Expand the `xps_hwicap_0` instance.

1-4-3. Click the drop-down button of the `ICAP_Clk`.

1-4-4. Select `clk_83_3333MHz`.

The connection appears as shown in Figure 6.

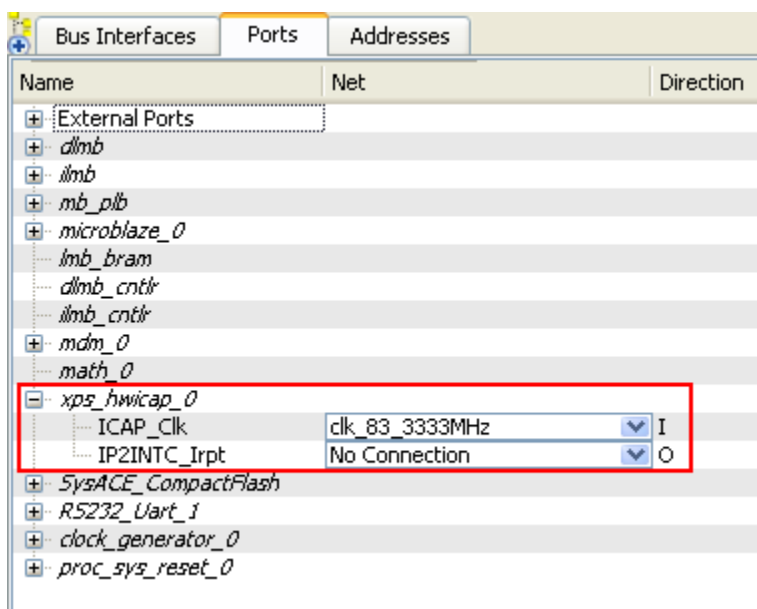


Figure 6: Connecting Clock Source to ICAP

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 different speed. In this tutorial, the system clock is also 83.33 MHz and hence, we are running the entire design in a single clock domain.

1-5. Generate netlist.

1-5-1. 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: Create a Software Project

Step 2

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

2-1. Export hardware design to SDK and create a board support package.

Be sure to add `xilfatfs` library support.

2-1-1. In XPS, select **Project > Export Hardware Design to SDK** to launch SDK.

2-1-2. Uncheck Include bitstream and BMM file.

2-1-3. Click **Export & Launch SDK**.

2-1-4. In SDK, select **File > New > Xilinx Board Support Package**.

2-1-5. Notice that the default Project Name is `standalone_bsp_0` and the OS is `standalone`.

2-1-6. Click **Finish** with default settings.

The Board Support Package Settings window opens.

2-1-7. Check the `xilfatfs` check box to select the FAT file system support for the Compact Flash card.

	Name	Version	Description
<input type="checkbox"/>	lwip130	2.00.a	lwIP TCP/IP Stack library: lwIP v1.3.0, Xilinx adapter v2....
<input checked="" type="checkbox"/>	xilfatfs	1.00.a	Provides read/write routines to access files stored on a F...
<input type="checkbox"/>	xilflash	2.01.a	Xilinx Flash library for Intel/AMD CFI compliant parallel flash
<input type="checkbox"/>	xilisf	2.01.a	Xilinx In-system and Serial Flash Library
<input type="checkbox"/>	xilmfs	1.00.a	Xilinx Memory File System

Figure 7: Selecting File System Support

2-1-8. Click **OK** to accept the settings and close the form.

2-2. Create a Make C Application Project.

2-2-1. In the Project Explorer view, select `standalone_bsp_0`.

2-2-2. Right-click and select **New > Project**.

2-2-3. Select **Xilinx C Project**.

2-2-4. Click **Next**.

2-2-5. Type `TestApp` as the Project Name.

2-2-6. Select **Empty Application** in the Project Application Template pane.

2-2-7. Click **Next**.

2-2-8. Select **Target an Existing Board Support Package**.

2-2-9. Click **Finish**.

2-3. Generate a Test Application.

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

This compiles the source files and generates `TestApp.elf` in the `reconfig_peripheral_lab\edk\SDK\SDK_Workspace_35\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, that 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.

2-4. Generate a Linker Script.

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

- 2-4-1. In SDK, select **Xilinx Tools > Generate linker script**.
- 2-4-2. Change the Heap size to **0x800** and the Stack size populates with **2048**.

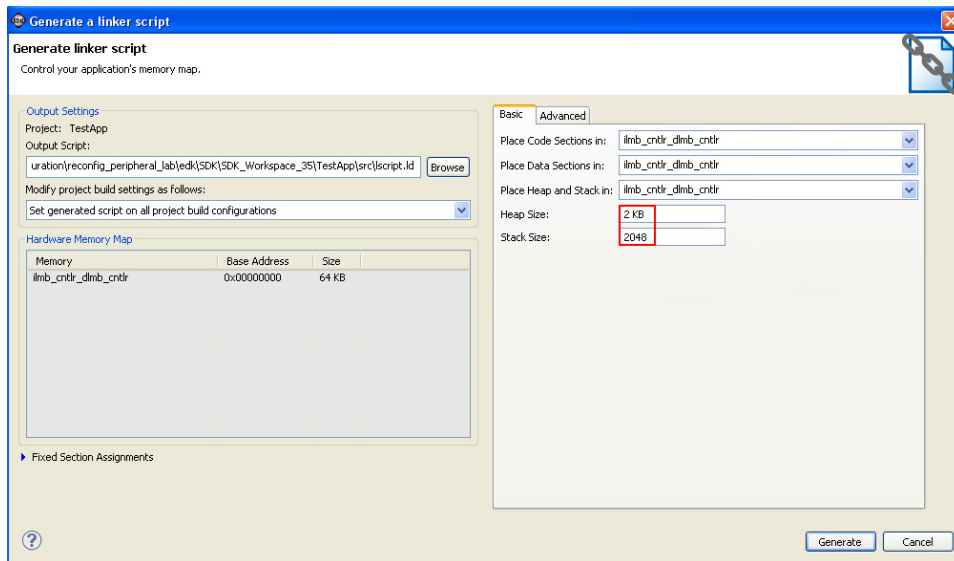


Figure 8: Generating a Linker Script

- 2-4-3.** Click **Generate**.
- 2-4-4.** Click **Yes** to overwrite the existing copy and recompile the application again.
- 2-4-5.** Select **File > Exit** to close SDK.

Step 3: Create a PlanAhead Project

Step 3

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.

3-1. Create a PlanAhead project and import the generated netlist files.

3-1-1. To open PlanAhead, select **Start > Programs > Xilinx ISE Design Suite 12.2 > PlanAhead > PlanAhead**.

3-1-2. Click **Create New Project**.

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

3-1-4. Browse to and select the `reconfig_peripheral_lab` directory for the Project location.

3-1-5. Click **Select**.

3-1-6. Set the Project name to **PlanAhead** in the New Project wizard.

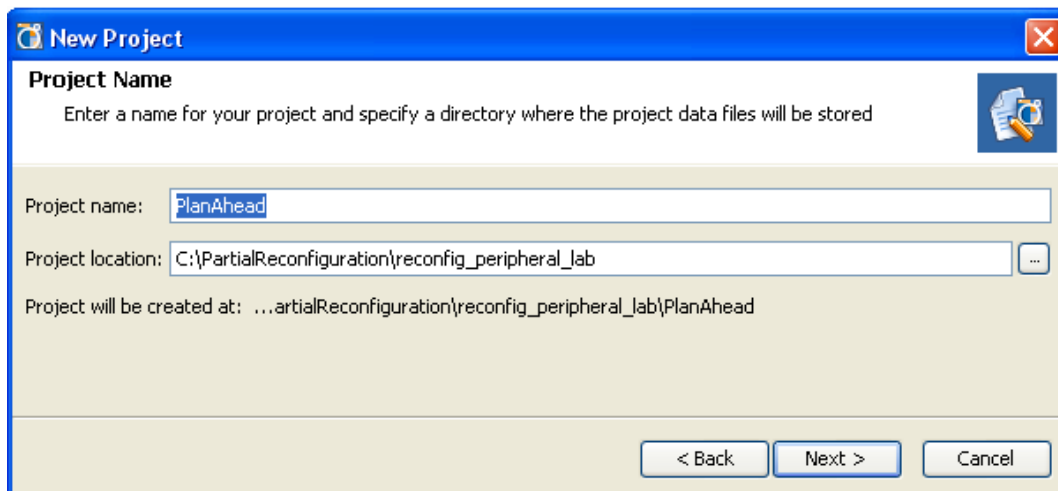


Figure 9: Project Name Page of the New Project Wizard

3-1-7. Click **Next**.

3-1-8. In the New Project Design Sources page, select **Specify synthesized (EDIF or NGC) netlist**.

3-1-9. Check the **Set PR Project** option.

3-1-10. Click **Next**.

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

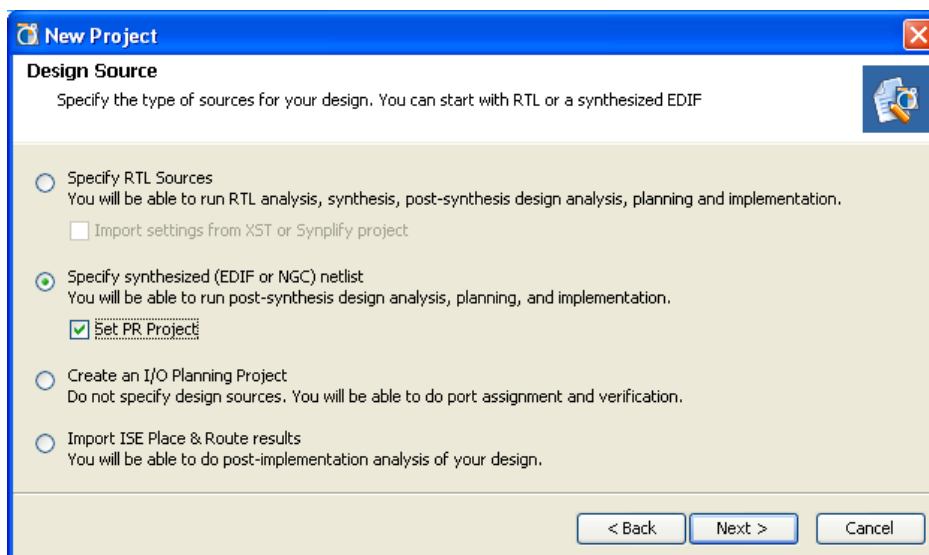


Figure 10: Importing Synthesized Netlists

3-1-11. Browse to **reconfig_peripheral_lab\edk\implementation**.

3-1-12. Select the **system.ngc** file and click **Open**.

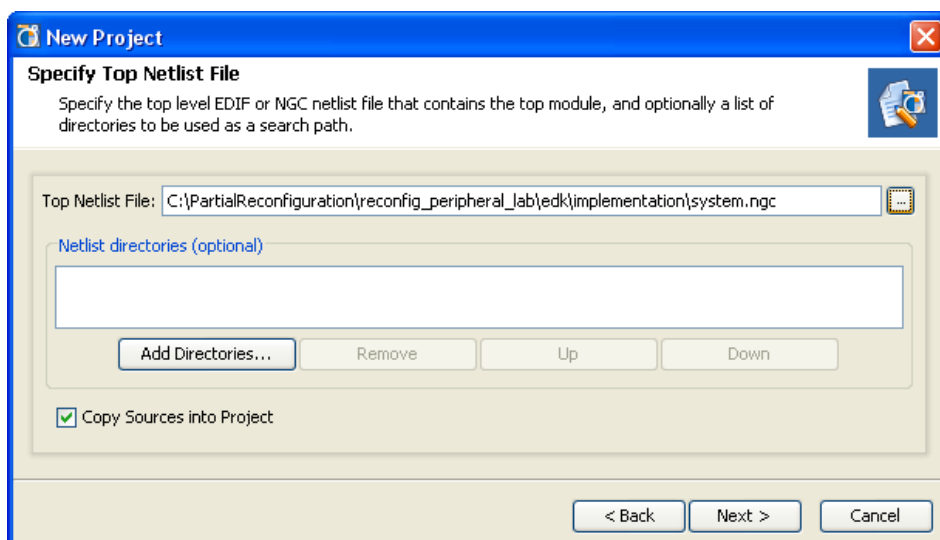


Figure 11: Selecting the Top-level Netlist File

3-1-13. Click **Next**.

The Constraint files (optional) page opens.

3-1-14. Click **Add Files**.

3-1-15. Browse to **\reconfig_peripheral_lab\edk\data**.

3-1-16. Select **system.ucf**.

3-1-17. Click **Open**.

3-1-18. Click **Next** to open the Product Family and Default Part page.

3-1-19. 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.

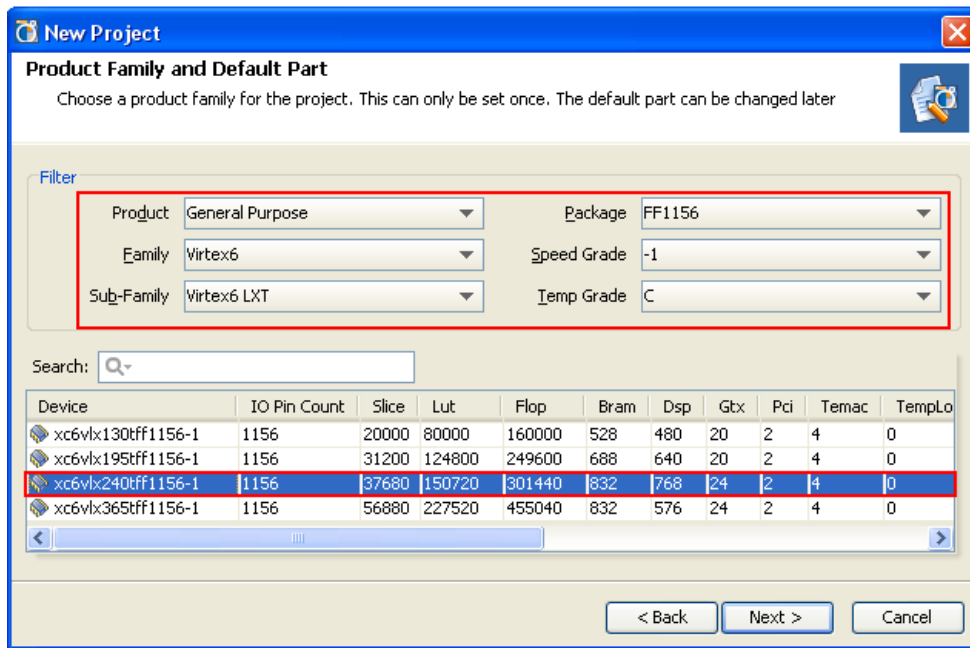


Figure 12: Selecting the Target Device

3-1-20. Click **Next**.

3-1-21. Click **Finish**.

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

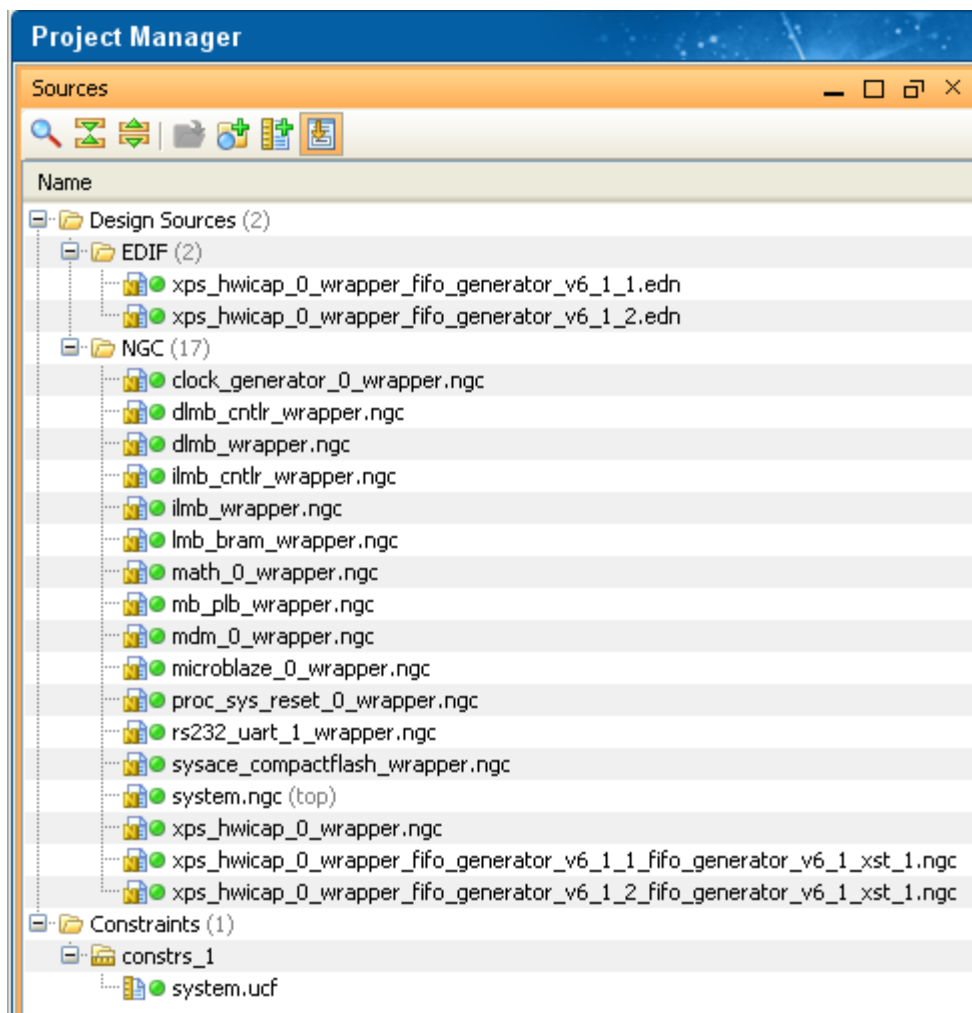


Figure 13: Design Hierarchy in PlanAhead

Step 4: Define a Reconfigurable Partition

Step 4

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

4-1. Define a Reconfigurable Partition (RP) with a black box reconfigurable module (RM).

4-1-1. 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.

4-1-2. Click **OK**.

4-1-3. Expand `math_o` instance by clicking on + button.

4-1-4. Select and right-click `math_0/USER_LOGIC_I/rp_instance` in the Netlist tab.

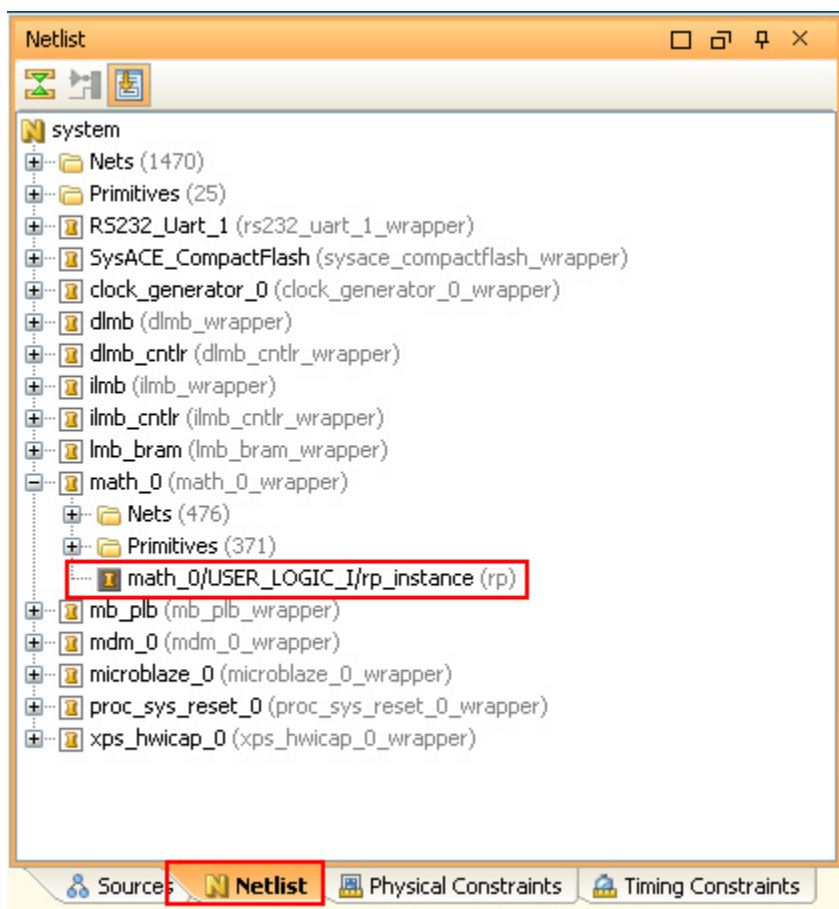


Figure 14: Netlists Hierarchy View

4-1-5. Select **Set Partition**.

4-1-6. Click **Next** twice.

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

4-1-7. Select **Add this Reconfigurable module as a black box without a netlist**.

4-1-8. Enter **math_BB** as the RM name since the partition does not yet have a defined netlist.

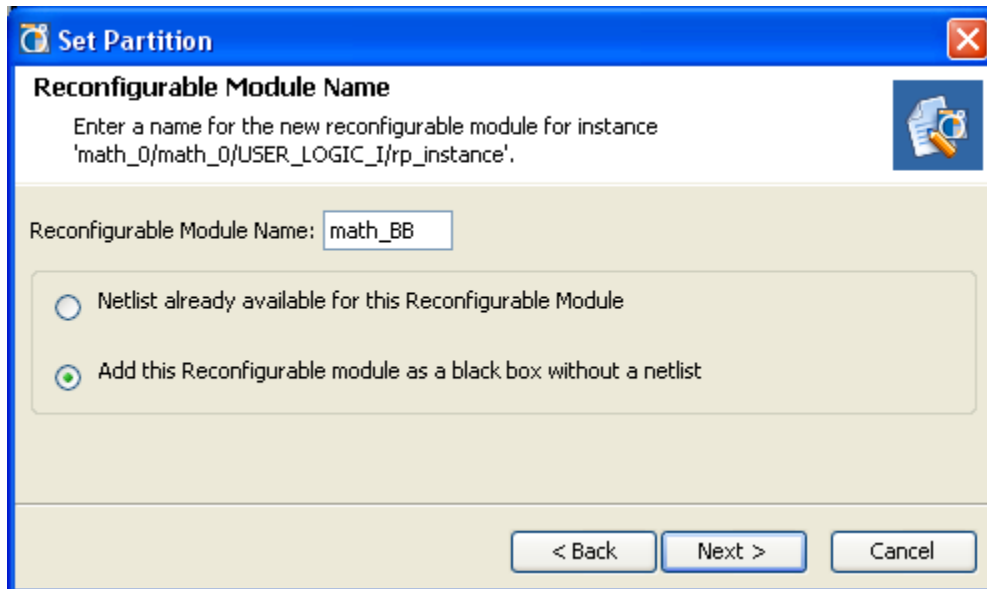


Figure 15: Setting a Partition

4-1-9. Click **Next**.

4-1-10. Click **Finish**.

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

Step 5: Add Reconfigurable Modules

Step 5

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

5-1. Add two reconfigurable modules: adder and multiplier.

5-1-1. In the Netlist window, select and right-click **math_0/USER_LOGIC_I/rp_instance**.

5-1-2. Select **Add Reconfigurable Module**.

5-1-3. Click **Next**.

The Add Reconfigurable Module dialog box displays, as shown in Figure 16.

5-1-4. In the Reconfigurable Module Name field, type **adder**.

5-1-5. Verify that **Netlist already available for this Reconfigurable Module** is selected.

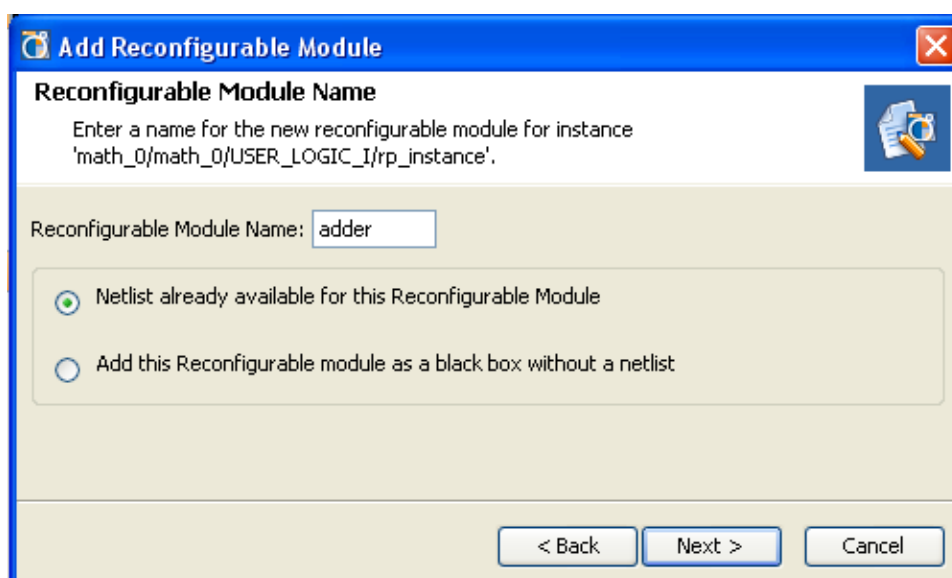


Figure 16: Adding a Reconfigurable Module

5-1-6. Click **Next**,

5-1-7. Browse to `reconfig_peripheral_lab\resources\Math\adder`, and select the **rp.ngc** file.

5-1-8. Click **Open**.

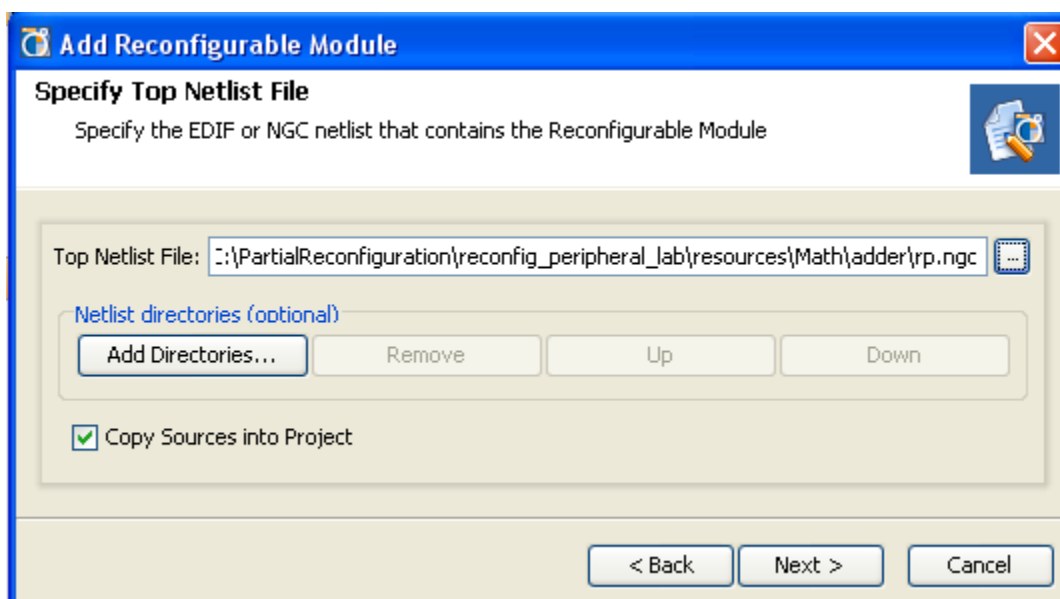


Figure 17: Locating the adder Version of math.ngc

5-1-9. Click **Next** twice.

5-1-10. Click **Finish**.

5-1-11. In the Netlist pane, expand Reconfigurable Modules hierarchy under `math_0/USER_LOGIC_I/rp_instance` to view the adder RM entry.

5-1-12. 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.

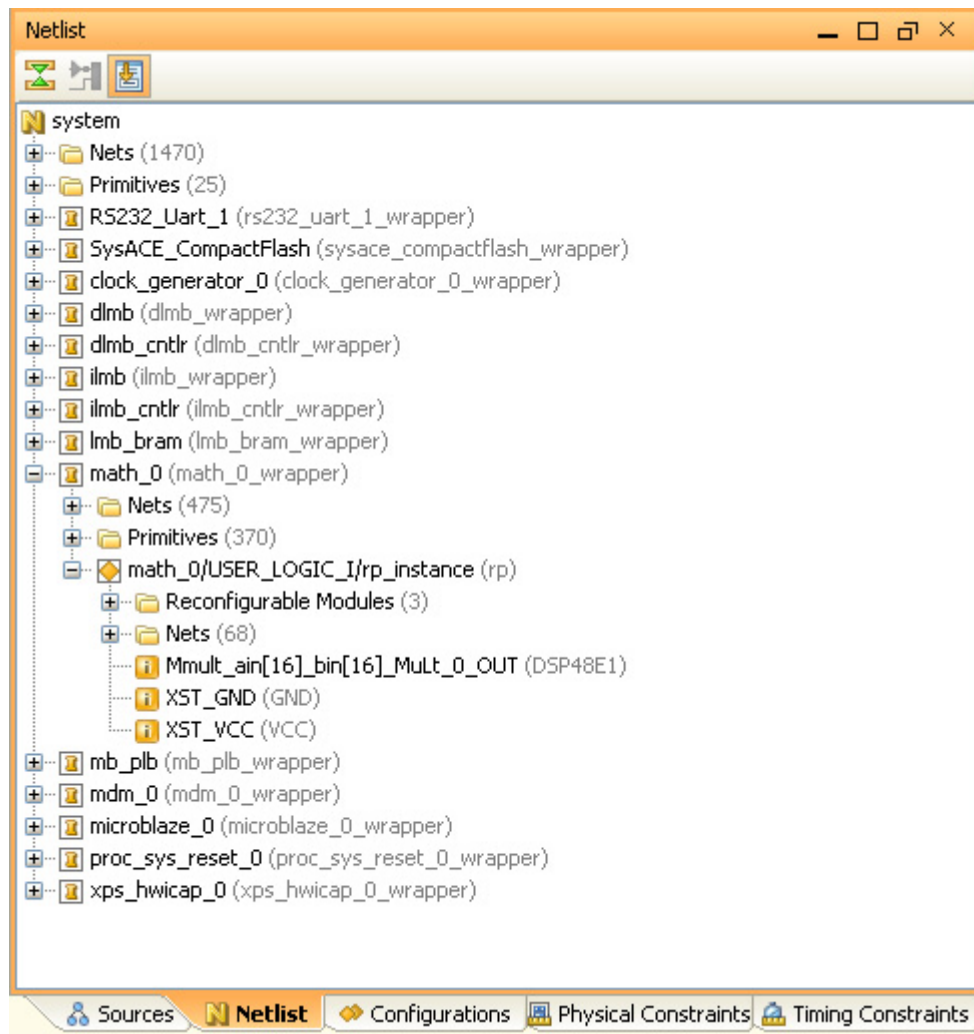


Figure 18: PlanAhead Project with adder and mult RMs Added

Step 6: Define the Reconfigurable Partition Region

Step 6

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.

6-1. Set the reconfigurable region.

6-1-1. In the Physical Constraints tab, select **pblock_math_0/USER_LOGIC_I/rp_instance**.

6-1-2. Right-click and select **Set Pblock Size**.

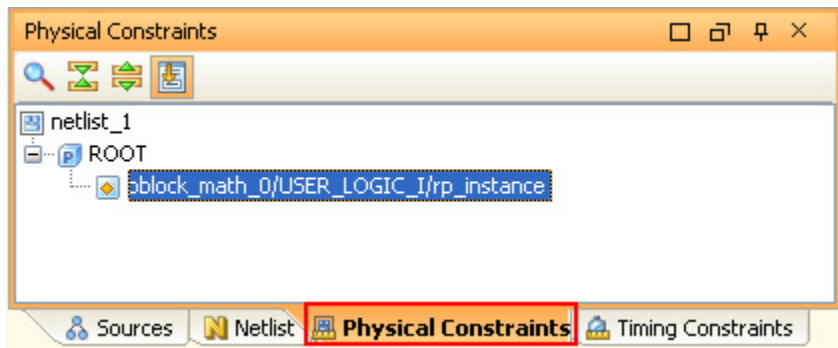


Figure 19: Setting Physical Constraints

6-1-3. Zoom to the top left quarter of the FPGA.

6-1-4. Move the cursor in the Device window.

6-1-5. Click and drag the cursor to draw a box that bounds SLICE_X10Y230:SLICE_X15Y239, 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.

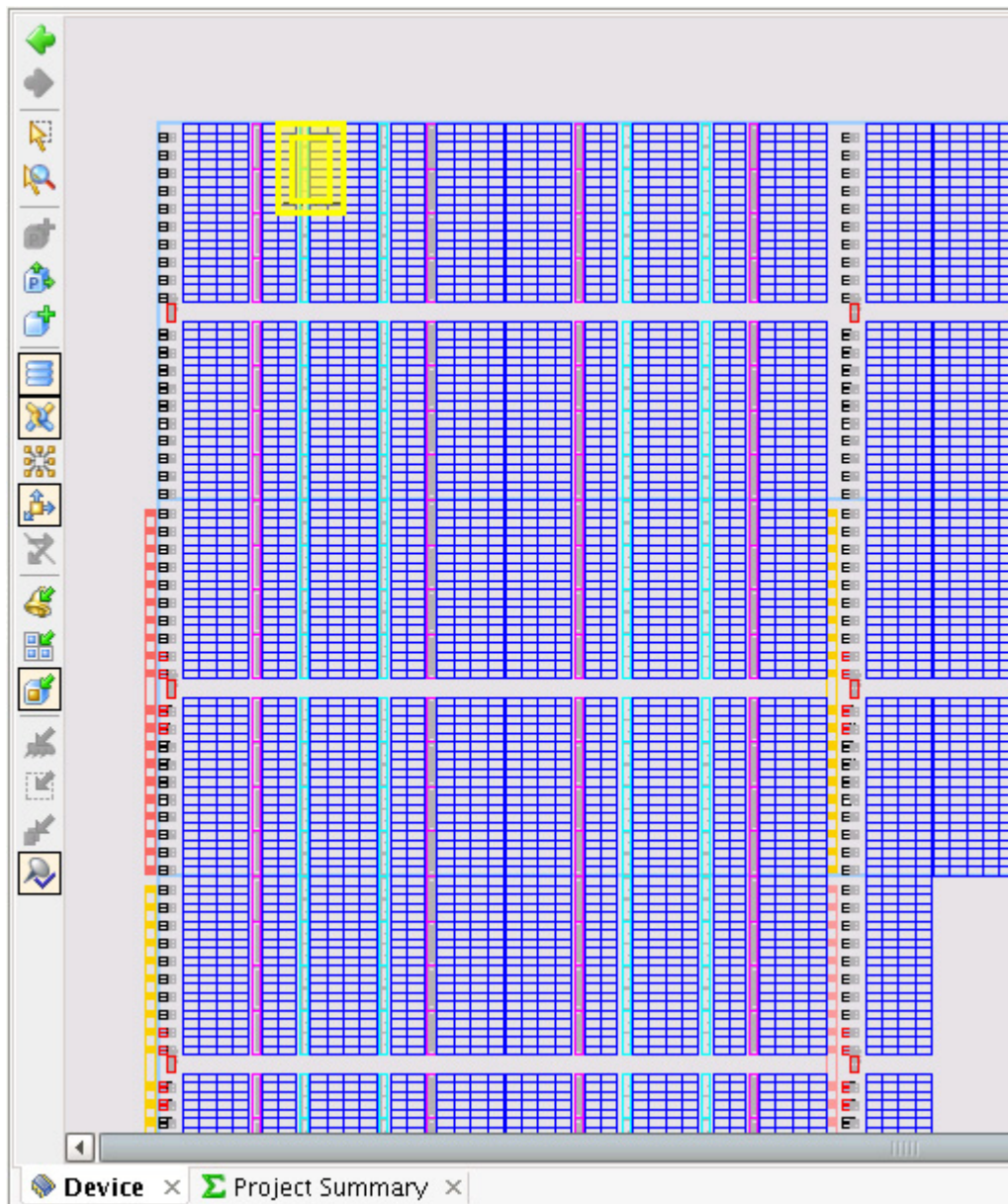


Figure 20: Closer View of the Pblock Area

- 6-1-6.** In the Set Pblock dialog box, verify that **SLICE** and **DSP48** are checked as the resources to be reconfigured, shown in the following figure.
- 6-1-7.** Click **OK**.

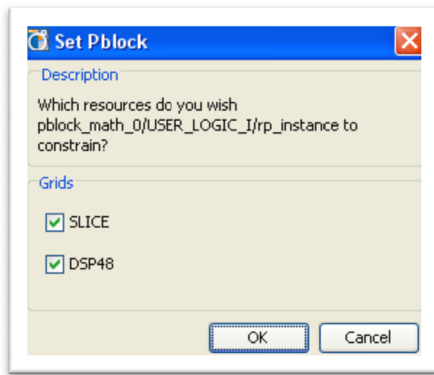


Figure 21: Setting Pblock with SLICE and DSP48

Step 7: Run Design Rule Checker

Step 7

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

7-1. Select and run PR-specific DRCs.

7-1-1. Select **Tools > Run DRC**.

7-1-2. Deselect **All Rules**.

7-1-3. Select **Partial Reconfig**.

7-1-4. Click **OK** to run the PR-specific design rules.

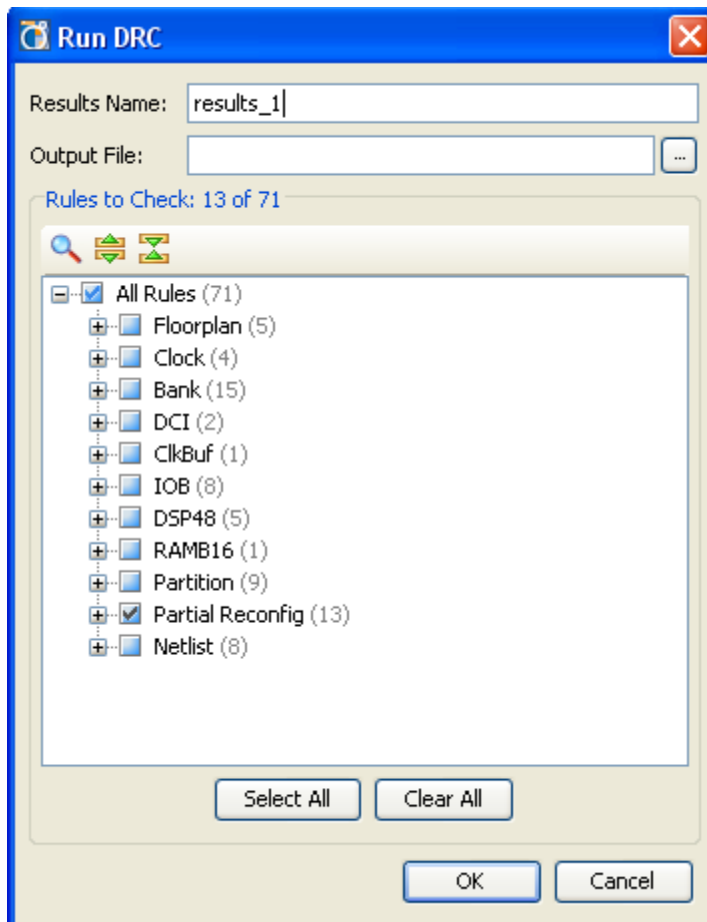


Figure 22: Running Design Rule Checks

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

Step 8: Create First Configuration, Implement, and Promote Step 8

Now you can create and implement the first Configuration.

8-1. Create a new strategy.

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

8-1-1. Select **Tools > Options**.

8-1-2. Select **Strategies** in the left pane.

8-1-3. Select **ISE 12** in the **Flow** drop-down box.

8-1-4. Under PlanAhead Strategies, select **ISE Defaults**.

8-1-5. Click the **+** button to create a new strategy.

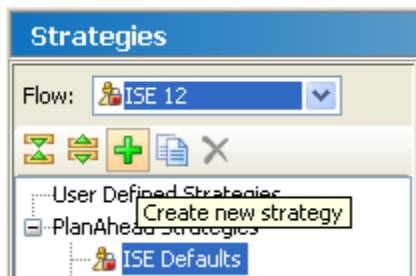


Figure 23: Creating a New Strategy

8-1-6. Name the new strategy **ISE12_BM**.

8-1-7. Click **OK**.

8-1-8. Under Translate (ngdbuild), click the **-bm** switch drop-down button.

8-1-9. Browse to and select the `reconfig_peripheral_lab\edk\implementation\system.bmm` file.

8-1-10. Click **Open**.

8-1-11. Click **OK**.

8-2. Run the implementation using mult as a variant.

8-2-1. At the bottom of the PlanAhead GUI, select the **Design Runs** tab.

8-2-2. Select the **config_1** run.

8-2-3. In the Implementation Run Properties window, select the **General** tab.

8-2-4. In the Name field, type **mult** as the **run** name.

8-2-5. Click **Apply** to change the run name from `config_1` to **mult**.

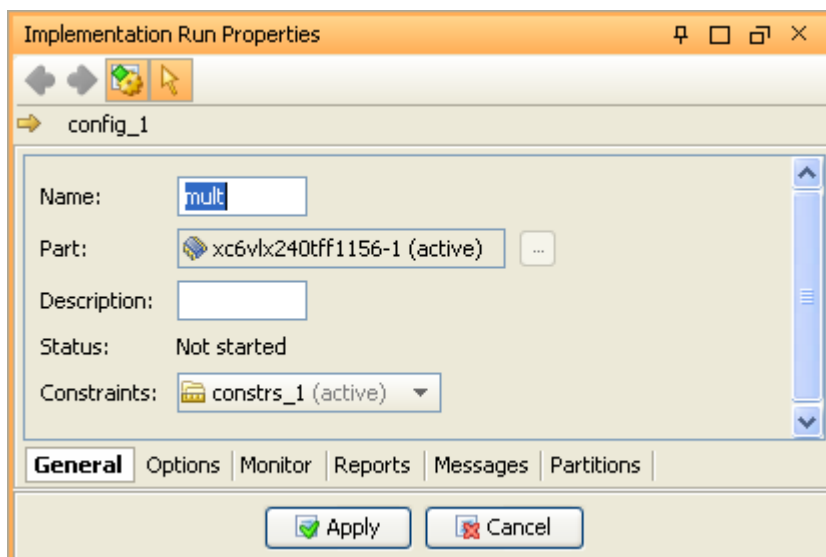


Figure 24: Implementation Run Properties View

8-2-6. In the Options tab, change the Strategy to ISE12_BM.

8-2-7. Click **Apply**.

8-2-8. In the Partitions tab, click the drop-down button in the Module Variant column and select **mult** as the variant, as shown in Figure 25.

8-2-9. Click **Apply**.

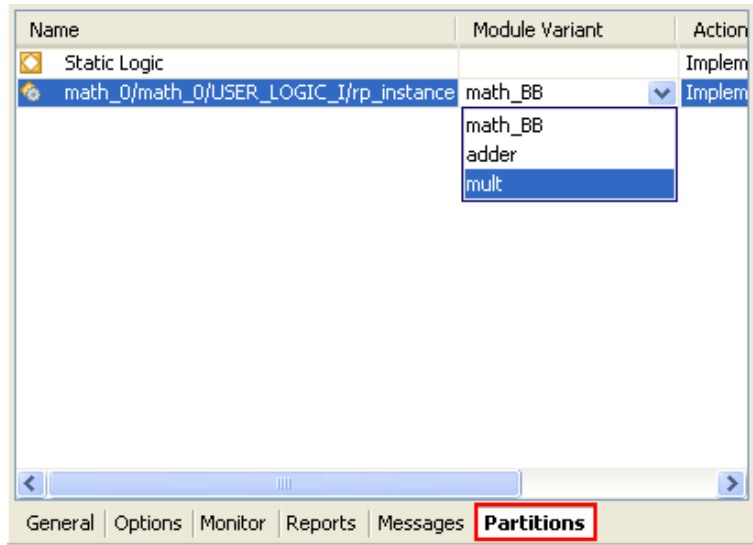


Figure 25: Implementation Run

8-2-10. In the Design Run window, select **mult**, and right-click and select **Launch Runs** to run the implementation.

8-2-11. Select **Launch Runs on Local Host**.

8-2-12. Click **OK**.

8-2-13. 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.

8-2-14. Select the **Promote Partitions** radio button, and click **OK**.

8-2-15. In the Promote Partitions dialog box, click **OK** to promote the current configuration so the implemented results are available for the subsequent configurations.

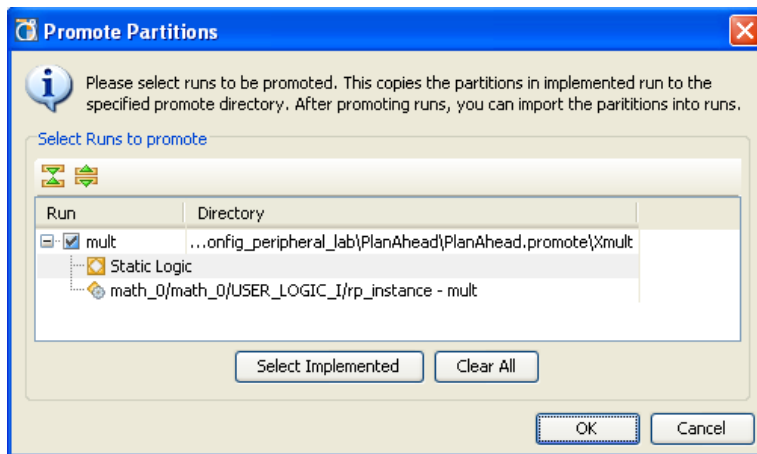


Figure 26: Promoting Partitions

Step 9: Create Other Configurations and Implement

Step 9

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.

9-1. Create multiple runs.

9-1-1. Select **Tools > Create Multiple Runs**.

The Create Multiple Runs window opens.

9-1-2. Click **Next** twice.

9-1-3. In the **Choose Implementation Strategies and Reconfigurable Modules** page, change the name of the configuration from `config_1` to **adder**.

9-1-4. Click **More**.

9-1-5. Change the name of `config_1` to **black_box**.

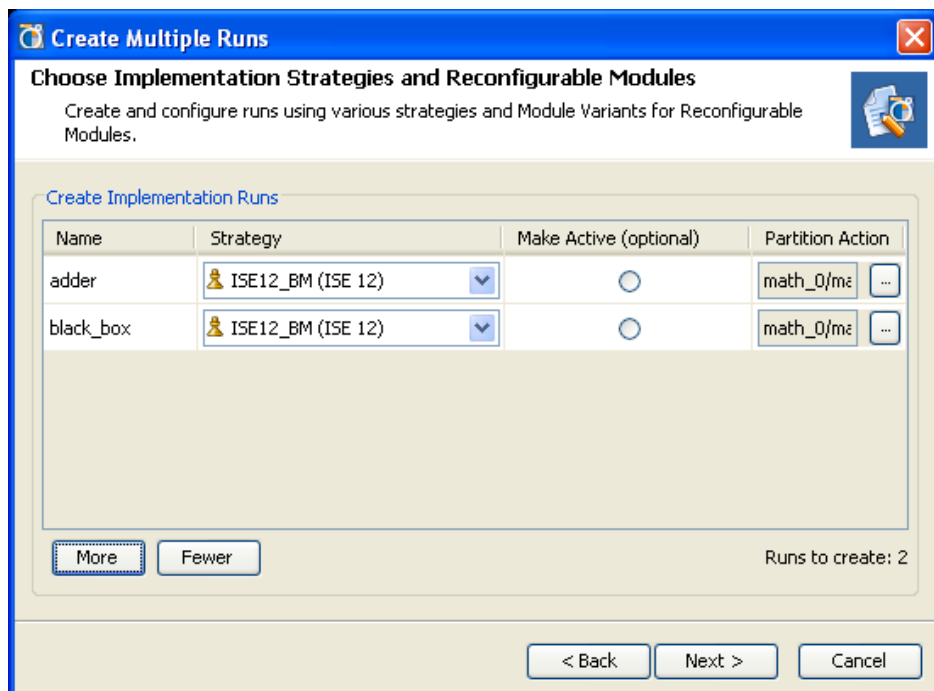


Figure 27: Creating Multiple Runs

9-1-6. In the adder configuration row, click the Partition Action field.

9-1-7. For the `rp_instance` row, click on drop-down arrow in Module Variant column, and select **adder** as the variant to be implemented (Figure 28).

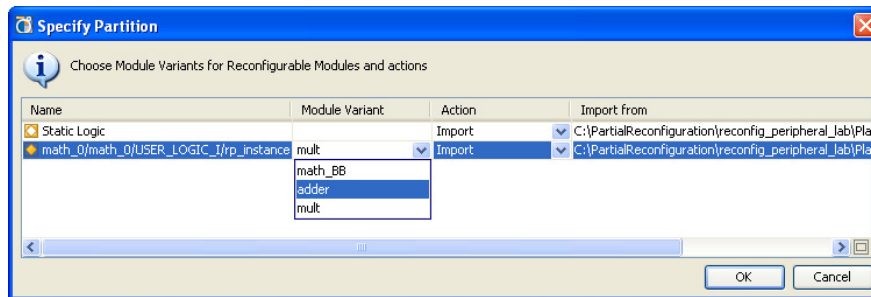


Figure 28: Selecting adder Module Variant

9-1-8. Click **OK**.

9-1-9. Similarly, select **math_BB** variant for the black_box run (row).

9-1-10. Click **Next**.

9-1-11. Select **Launch Runs on Local Host**.

9-1-12. Click **Next**.

9-1-13. Click **Finish** to run the implementations for both configurations.

Step 10: Run Partial Reconfiguration to Verify Utility

Step 10

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.

10-1. Run PR_Verify utility.

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

10-1-1. In the Configurations window, select any of the configurations.

10-1-2. Right-click and select **Verify Configuration**.

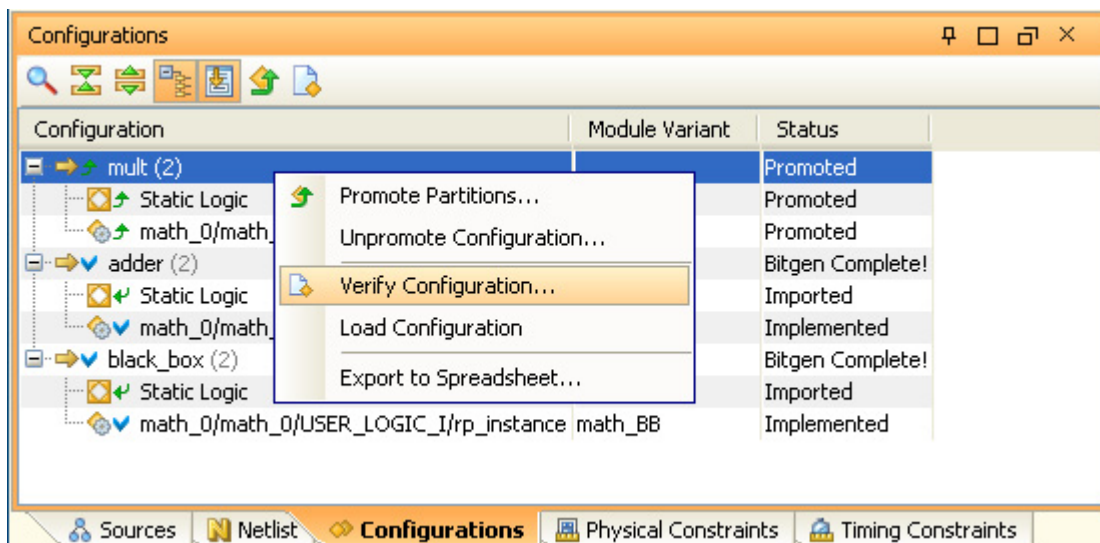


Figure 29: Verifying All Configurations

10-1-3. Press **Shift** and select all configurations.

10-1-4. Click **OK**.

10-1-5. The PR_Verify utility runs and reports that there were no errors.

Step 11: Generate Bit Files

Step 11

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

11-1. Generate full and partial bitstreams.

11-1-1. In the Design Runs window, press **Shift** and select the following three designs runs.

- **mult**
- **adder**
- **black_box**

11-1-2. 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.

11-1-3. Save the project.

11-1-4. Close PlanAhead.

Step 12: Create Image and Test

Step 12

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`.

12-1. Rename partial bitstream files and the generate system ace file.

12-1-1. Launch the EDK bash shell as follows:

- From XPS, select **Project > Launch Xilinx Bash Shell**, or
- From your Windows environment, select **Start > Programs > Xilinx ISE Design Suite 12.2 > EDK > Tools > Xilinx > Bash Shell**.

12-1-2. In the Bash shell, go to the `\reconfig_peripheral_lab\image` directory.

12-1-3. 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_Workspace_35/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.

12-1-4. 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
```

12-1-5. 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

12-2. Copy the system.ace and the three partial bit files on a compact flash memory card.

- 12-2-1. Place a blank Compact Flash memory card in a CompactFlash writer.
- 12-2-2. Using Windows Explorer, copy the three partial bit files and the `system.ace` file from `reconfig_peripheral_lab\image` folder to the CompactFlash card.
- 12-2-3. Place the CompactFlash card in the ML605 board.
- 12-2-4. Set the SACE Mode pins (S1) to 0111 (dn-up-up-up) to configure the FPGA device from the CompactFlash.
- 12-2-5. Connect your PC to the ML605 with the provided USB cable.
- 12-2-6. Install the driver, if necessary. For instructions, see the *ML605 Hardware User Guide*:
http://www.xilinx.com/support/documentation/boards_and_kits/ug534.pdf.
- 12-2-7. Start a HyperTerminal window, connecting using COMx at 115200 baud and power ON the ML605 board.
- 12-2-8. Press **CPU Reset**.
- 12-2-9. 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.