Partial Reconfiguration of a Processor Tutorial

PlanAhead Design Tool

UG744 (v14.6) June 19, 2013

XILINX®

Revision History

Date	Version	Revision
06/19/2013	14.6	Validated with release.
04/25/2013	14.5	Validated with release.
04/24/2012	14.1	Validated with release. Updated HWICAP version.

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 that includes a lower-level module defining one Reconfigurable Partition (RP) and two Reconfigurable Modules (RM). The two RM perform addition and multiplication functions. You will use SDK to create a software application that 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

The FPGA design in this tutorial is targeted to the Virtex®-6 FPGA ML605 Evaluation Board. The design targets a Virtex-6 xc6vlx240tff1156-1 device.

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.

Understanding the Processor System

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

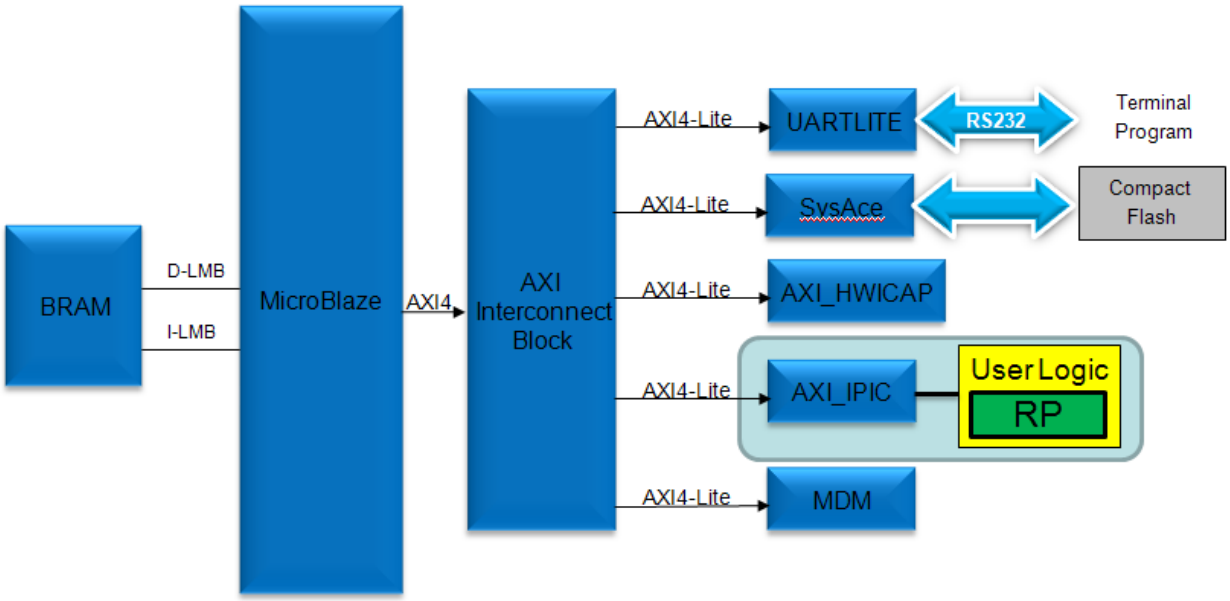

Figure 1: Top-Level Design

Software Requirements

The PlanAhead tool is installed with ISE Design Suite software. Before starting the tutorial, be sure that the PlanAhead tool is operational, and that the tutorial design data is installed.

IMPORTANT: *Partial Reconfiguration is a licensed feature of the PlanAhead tool. Completing this tutorial requires a valid license to access the Partial Reconfiguration feature.*

For installation instructions and information, see the *ISE Design Suite 14: Release Notes, Installation, and Licensing (UG631).*

Hardware Requirements

Xilinx recommends a minimum of 2 GB of RAM when using the PlanAhead tool on larger devices. For this tutorial, a smaller design is used, and the number of designs open at one time is limited. Although 1 GB is sufficient, it can impact performance.

Preparing the Tutorial Design Files

This tutorial uses a reference design, **UG744_design_files.zip**, which must be unzipped to a write-accessible directory. You can download a copy of the reference design from:

http://www.xilinx.com/cgi-bin/docs/rdoc?v=14.6;t=planahead+tutorials

Extract the zip file contents into any write-accessible location which will be referred to in this tutorial as the extraction directory, or <*Extract_dir*>.

RECOMMENDED: *The tutorial sample design data is modified while performing this tutorial. A new copy of the original PlanAhead_Tutorial data should be extracted each time you run this tutorial.*

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

Project Directory Structure

The directory structure is:

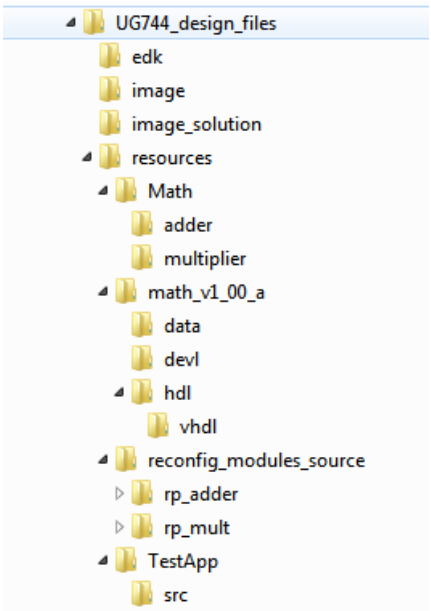

Figure 2: The Project Directory

- The `/edk` directory is used to create a processor system.

- 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.
- 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) provides:
    - The necessary processor bus connections
    - The required peripheral services (in this case, one slave register and a software reset), and
    - A placeholder for the math functionality module

Lab 1: Partial Reconfiguration of a Processor

Step 1: Creating a Processor Hardware System

1. Select **Start > All Programs > Xilinx Design Tools > Xilinx Design Suite 14.6 > 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: `<Extract_Dir>/edk`
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.

1. In Board and System Selection form, select **Xilinx** as a Board Vendor.
2. In Board Name field, select Virtex-6 ML605 Evaluation Platform.
3. In Board Revision field, select **D**.
4. Click **Next** with other default options selected.
5. Select 50.00 MHz from the Processor Frequency drop-down menu.
6. Select 64 KB from the Local Memory Size drop-down menu.
7. In the selected peripherals list on the right, remove all devices *except*: **RS232_Uart_1** and **SysACE_CompactFlash**.
8. Click **RS232_Uart_1** and configure it with a baud rate of **115200**.
9. Click **Finish**.
10. If the **Next Step** dialog box opens, click **OK** to start using Platform Studio and open the System Assembly View window as shown in Figure 3.

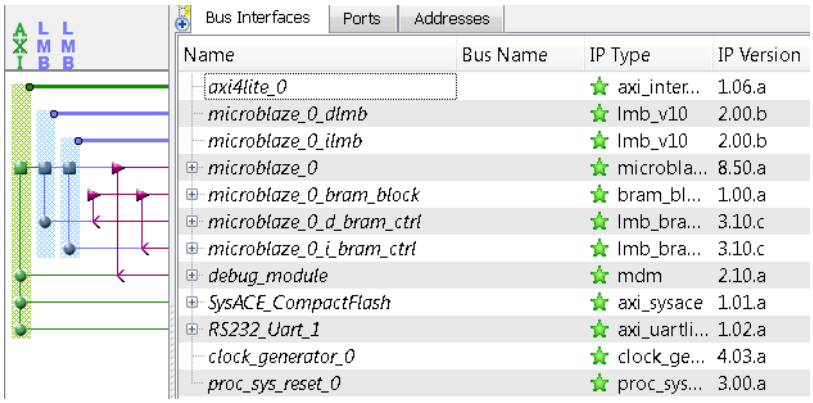

Figure 3: Displaying the System Assembly View

Partial Reconfiguration Design Details

1. Copy the *<Extract_Dir>*/resources/math_v1_00_a folder to ./edk/pcores
2. In <Extract_Dir>/edk/pcores/math_v1_00_a/hdl/vhdl, examine the user_logic.vhd file.
   - At line 133, and at line 158, it declares a component that will be used in the reconfigurable partition.
   - 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:
3. Select **Edit > Preferences > ISE Text Editor**.
4. Click to select the Show line numbers check box.
5. Click **Apply** and then **OK**.
6. 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.
7. Expand the USER folder.
8. Select **MATH**.

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

   A properties form opens.

10. Click **OK** twice to add the IP with the default settings and connect it to the microblaze_0 instance.

11. In the IP Catalog tab, select the FPGA Internal Configuration Access Port (v2.03.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.

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

    Note: When 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 **50.0000** for the ICAP_Clk of the `axi_hwicap_0` instance, select **<AUTO>** under the source column, and click **OK**.
5. Click **OK** to close the form.

   The connection appears as shown in Figure 4.

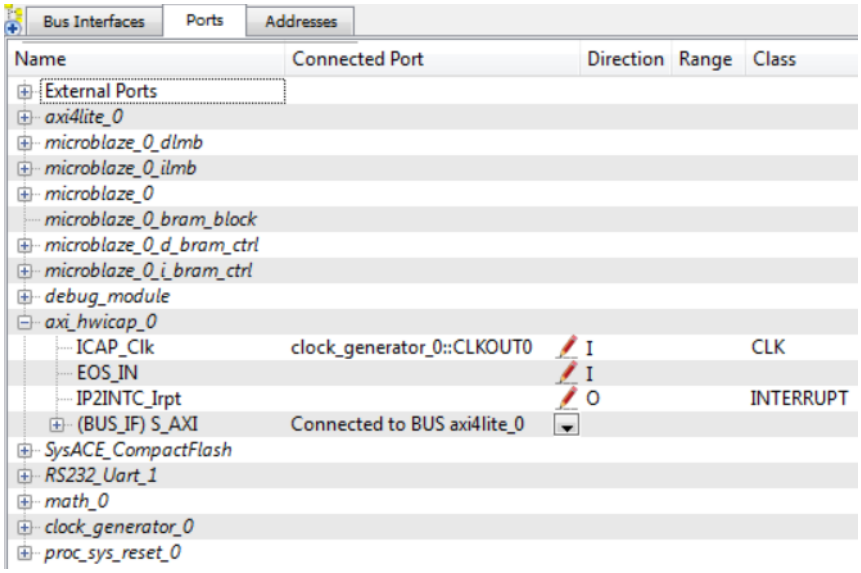

Figure 4: Connecting Clock Source to ICAP

Partial Reconfiguration Design Details

The `axi_hwicap` pcore allows a 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 MHz and hence, we are running the entire design in a single clock domain.

Notice that there is EOS_IN port on the `axi_hwicap_0` instance. This port is available for the designer to connect to a separate signal that can be asserted only when the system is stable and the reconfiguration can be done, to take care of a situation where reconfiguration command may be issued before the system is stable. You can instantiate a STARTUP block and connect the port correctly and automatically by selecting configuration parameter **instantiate STARTUP primitive in the HWICAP core** option in the `hwicap` pcore configuration GUI.

1. Select the **Bus Interfaces** tab.
2. Double-click on the axi_hwicap_0 instance and click on the check box of **instantiate STARTUP primitive in the HWICAP core** option in the User tab.
3. Click **OK** to accept the settings.
4. Select the **Ports** tab and observe that EOS_IN port is not listed as it is connected to STARTUP block which is automatically instantiated with the selected option.

Generating Netlists

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

After 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, making 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 opens.
4. Browse to the `<Extract_Dir>/edk/SDK/SDK_Export` directory, and click **OK** to open SDK after importing hardware specification of the system.
5. In SDK, select **File > New > 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.

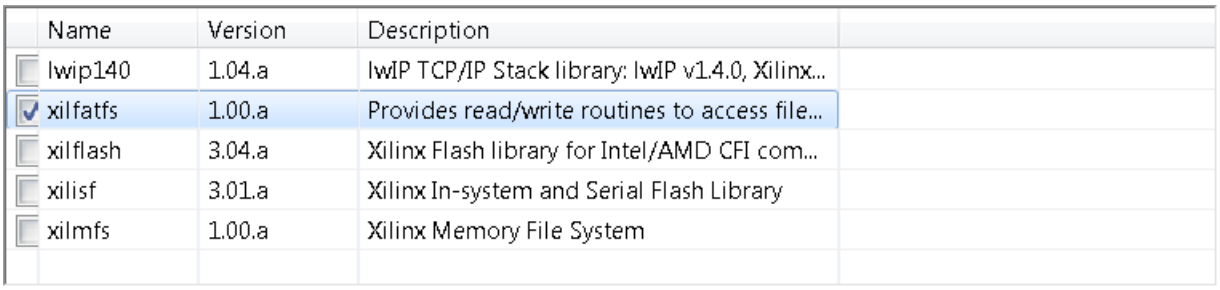

Figure 5: Selecting File System Support

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

Creating a Xilinx C Project

1. Select **File > New > Application Project**.
2. Type **TestApp** for the Project Name.
3. Select **Use Existing** option under the Board Support Package field and click **Next**.
4. Select **Empty Application** in the Project Application Template pane.
5. 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 `<Extract_Dir>/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 `<Extract_Dir>/edk/TestApp/Debug/` folder.

Partial Reconfiguration Design Details

Examine the `<Extract_Dir>/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

1. Be sure that the Heap and Stack sizes are set to **2048 (0x800)**.
2. In SDK Project Explorer view, select **TestApp**.
3. Right-click and select **Generate linker script**.
4. Change the Heap size and the Stack size to **2048**.

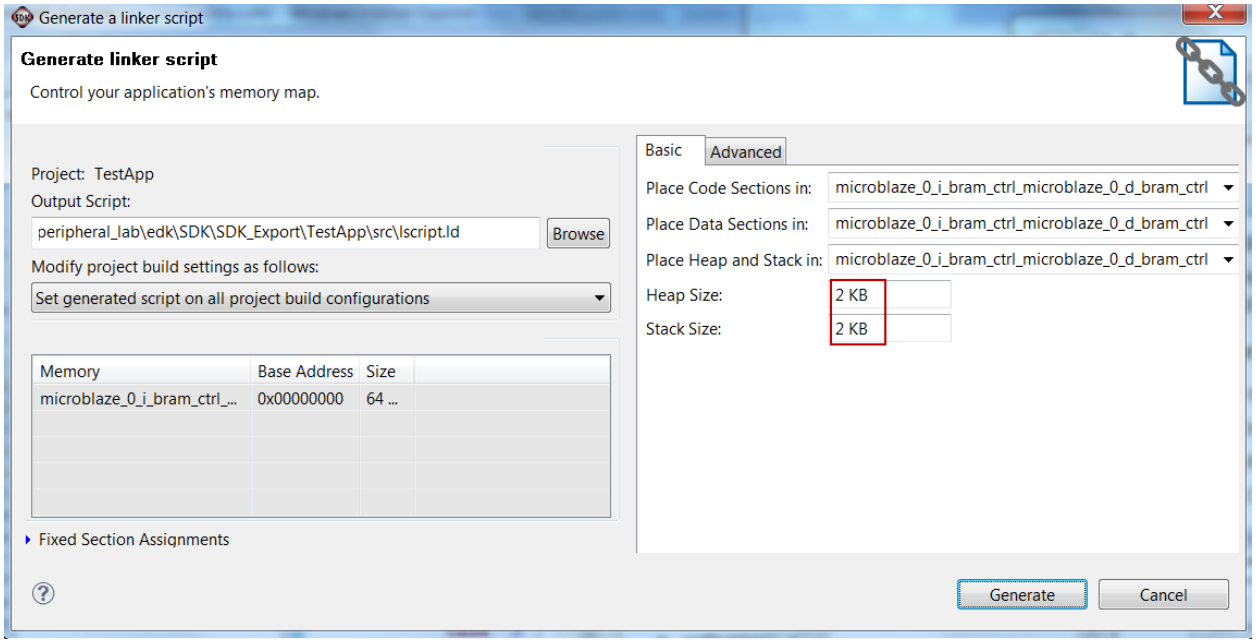

Figure 6: Generating a Linker Script

5. Click **Generate**.
6. Click **Yes** to overwrite the existing copy and recompile the application again.
7. Select **File > Exit** to close SDK.
8. In XPS, select **File > Exit** to close XPS.

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 > All Programs > Xilinx Design Tools > Xilinx ISE Design Suite 14.6 > PlanAhead > PlanAhead**.
2. Click **Create New Project**.
3. Click **Next**.
4. Browse to and select the `<Extract_Dir>/` directory for the Project location.
5. Click **Select**.
6. Type **PlanAhead** for the Project name in the New Project wizard.
7. Make sure that Create Project Subdirectory is checked.

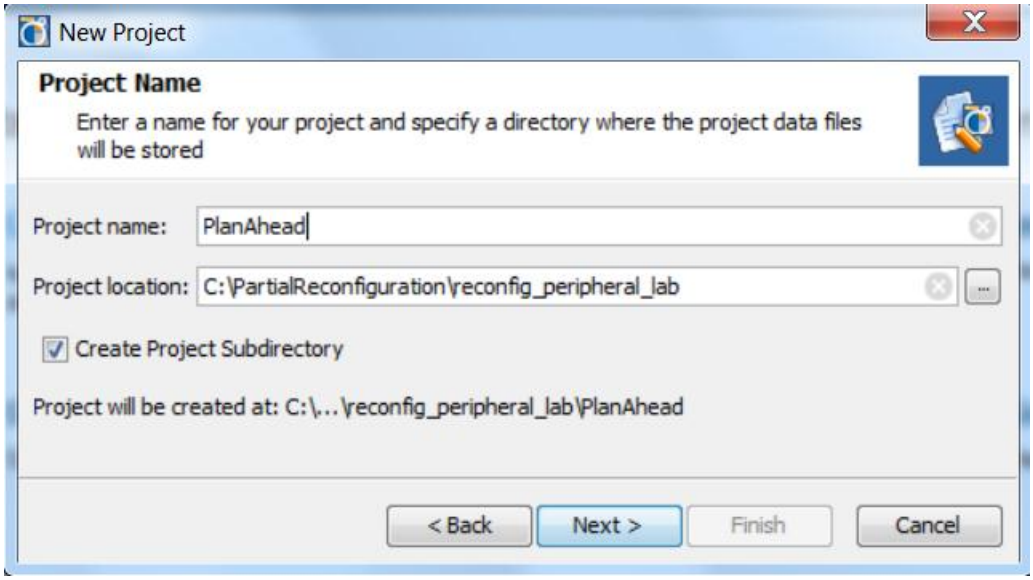

Figure 7: New Project Wizard

8. Click **Next**.
9. In the Project Type page, select **Post-synthesis Project**.
10. 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.

11. Click **Next**.

    Important! The Enable Partial Reconfiguration option is available only if you have a license for Partial Reconfiguration.

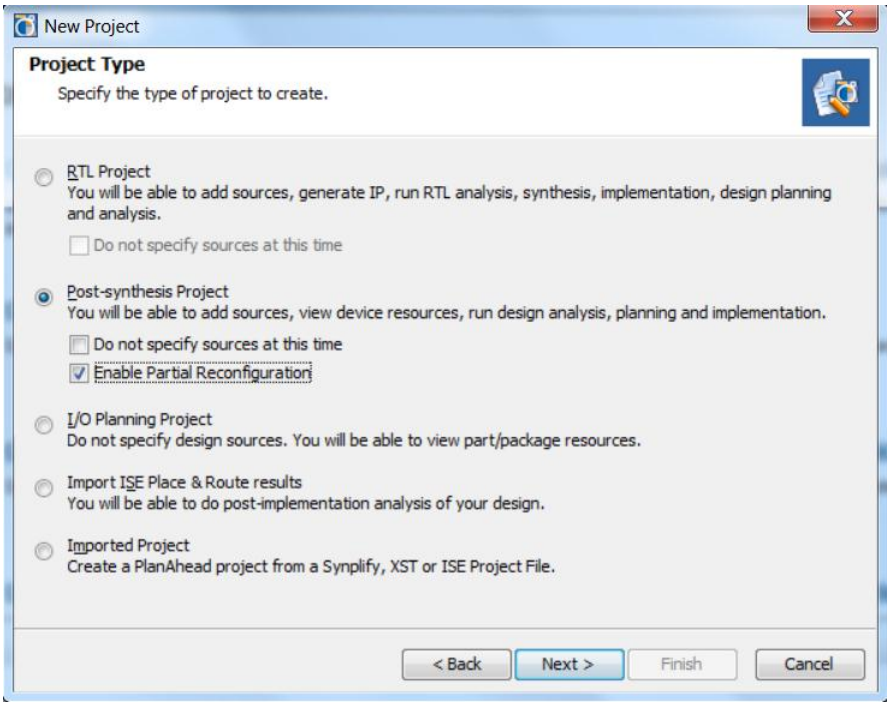

Figure 8: Importing Synthesized Netlists

12. Click the **Add Files** button**.**
13. Browse to the `<Extract_Dir>/edk/implementation/` folder.
14. Select all NGC files including the `system.ngc` file, and click **OK**.
15. In the Top column, click the radio button next to `system.ngc` to identify it as the top-level design file.

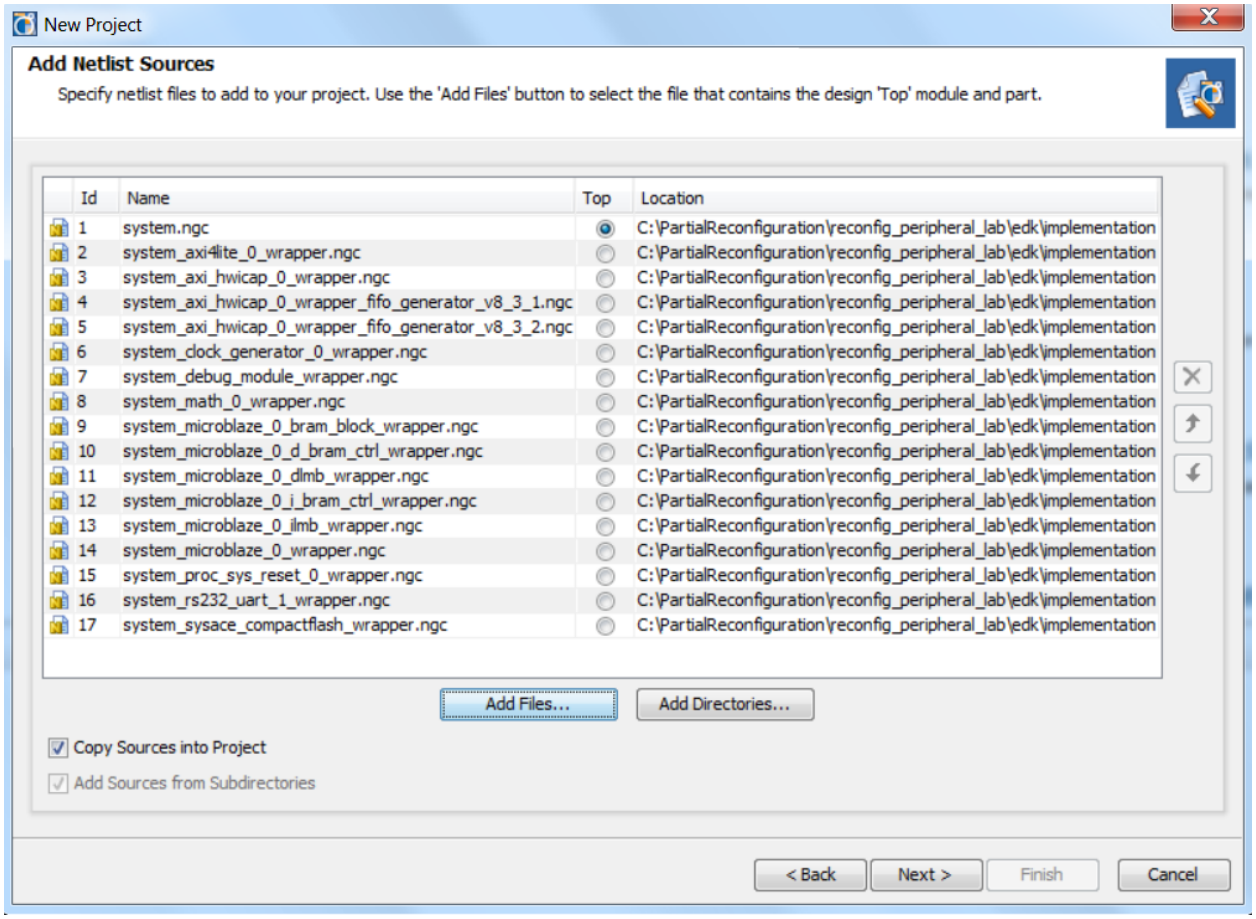

Figure 9: Selecting the Top-level Netlist File

16. Click **Next**.

    The Add Constraint files (optional) page opens.

17. Click **Add Files**.

18. Browse to `<Extract_Dir>/edk/data/`

19. Select `system.ucf`.

20. Click **OK**.

21. Click **Next** to open the Default Part page.

22. Click on the **Boards** under the **Specify** column, click on the drop-down button of the Family and select **Virtex-6**, and **select Virtex-6 ML605 Evaluation Platform** as shown in the following figure. Click on the **Boards** under the **Specify** column, click on the drop-down button of the Family and select **Virtex-6**, and **select Virtex-6 ML605 Evaluation Platform** as shown in the following figure.

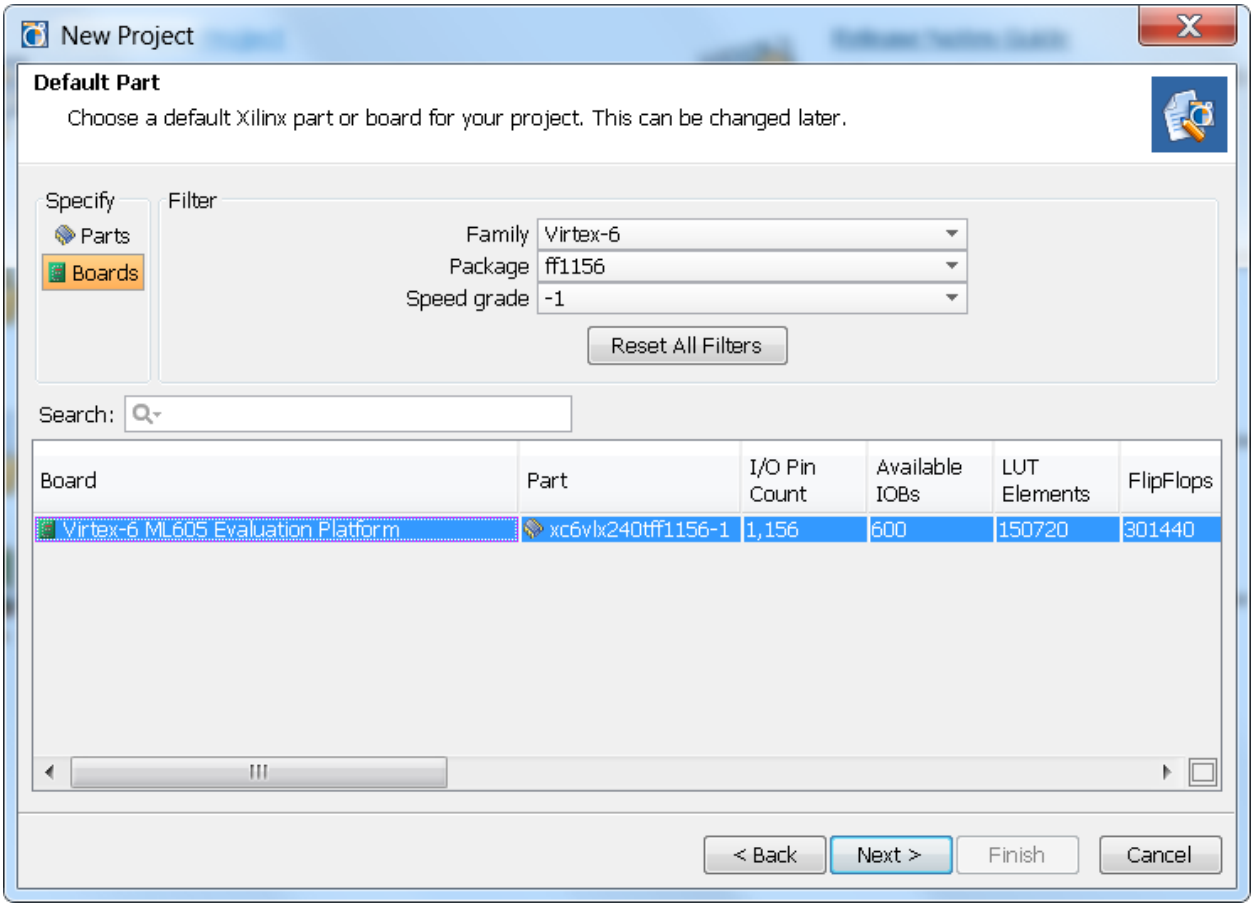

Figure 10: Selecting the Target Device

23. Click **Next**.
24. Click **Finish**.

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

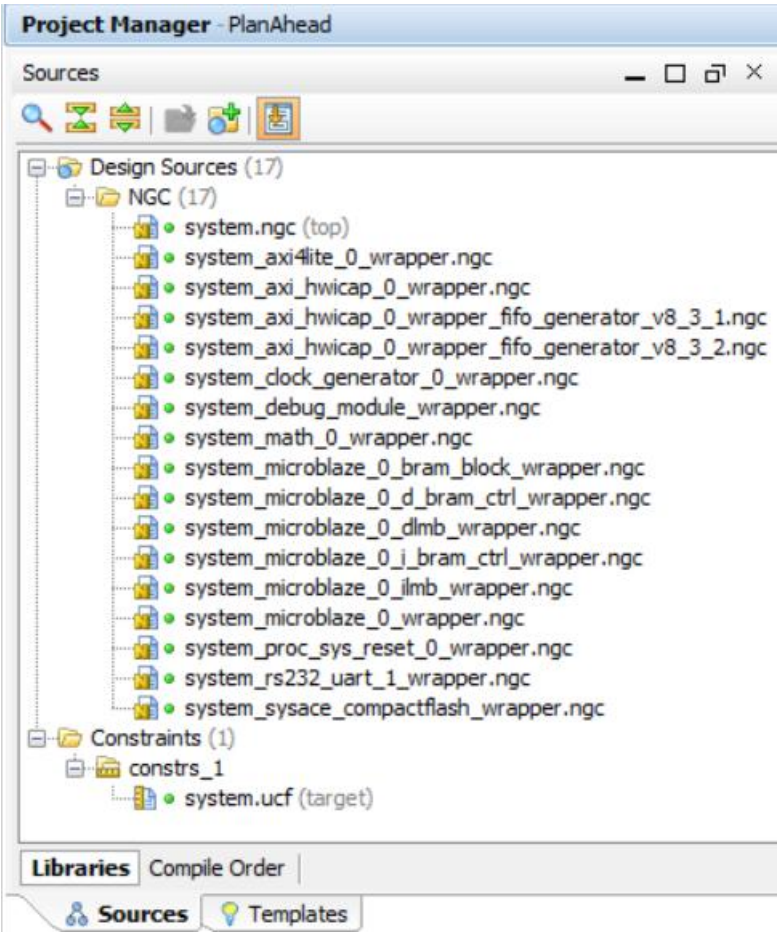

Figure 11: Design Hierarchy in PlanAhead

Step 4: Defining a Reconfigurable Partition

This design has one reconfigurable partition that you must explicitly define.

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

1. Click **Open Synthesized Design** under **Netlist Analysis** step of the Project Manager Flow Navigator pane to invoke the netlist files parser.

   This is necessary 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** twice.
3. Expand the math_0 instance.
4. Select **math_0/USER_LOGIC_I/rp_instance** in the Netlist view.

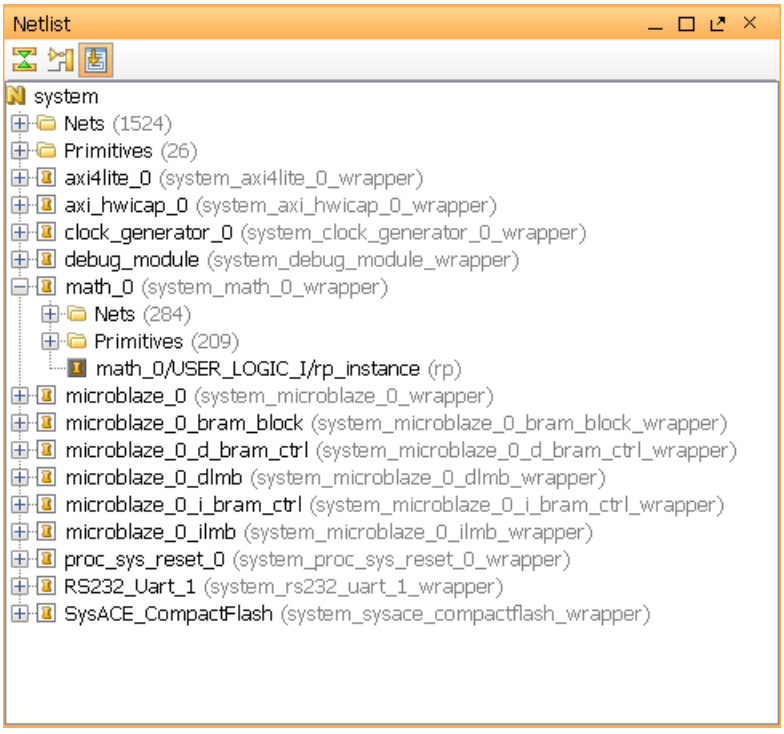

Figure 12: Netlists Hierarchy View

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

   The Set Partition dialog box opens.

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

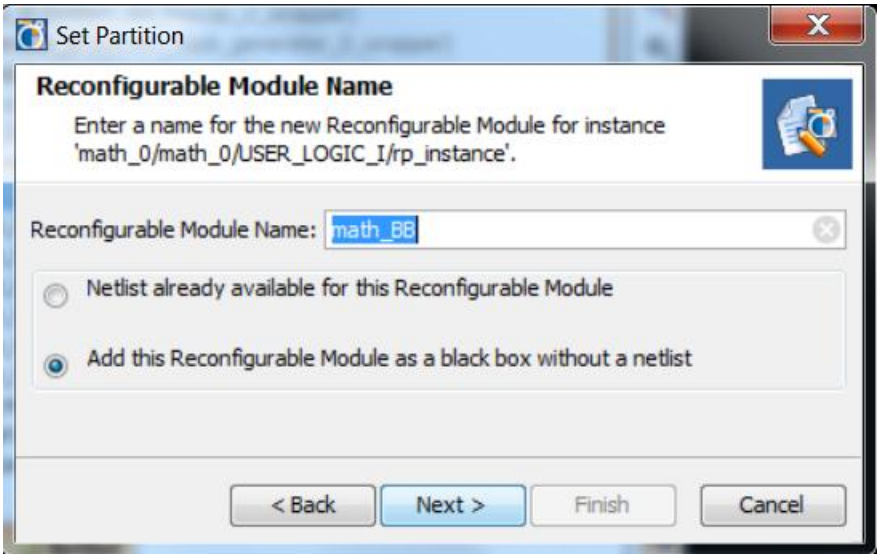

Figure 13: Setting a Partition

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 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**.

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

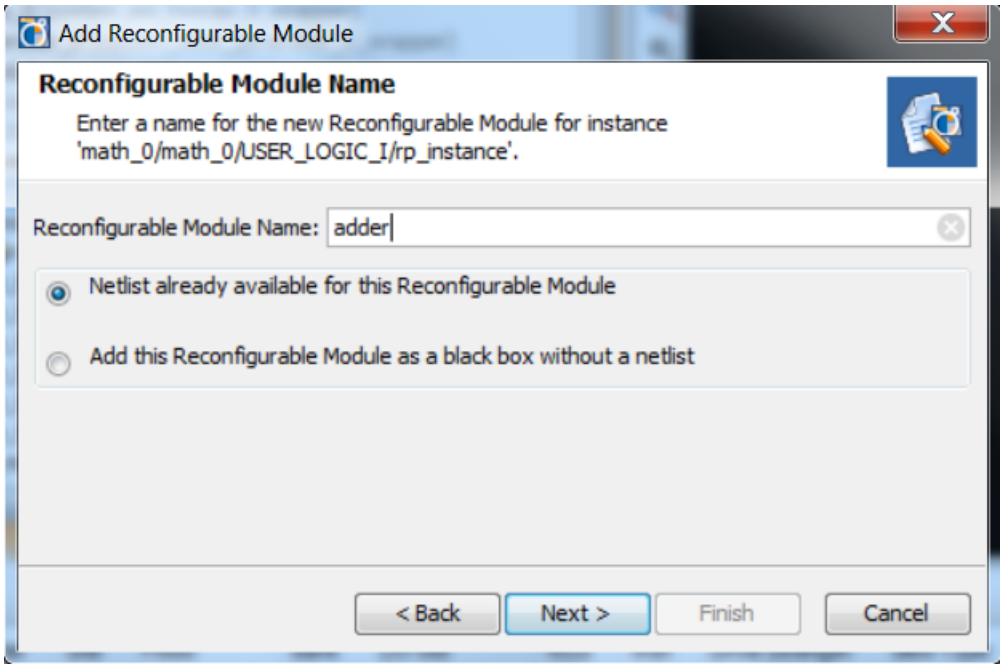

Figure 14: Adding a Reconfigurable Module

6. Click **Next**.

   Browse to `<Extract_Dir>/resources/Math/adder/` and select the `rp.ngc` file.
7. Click **OK**.

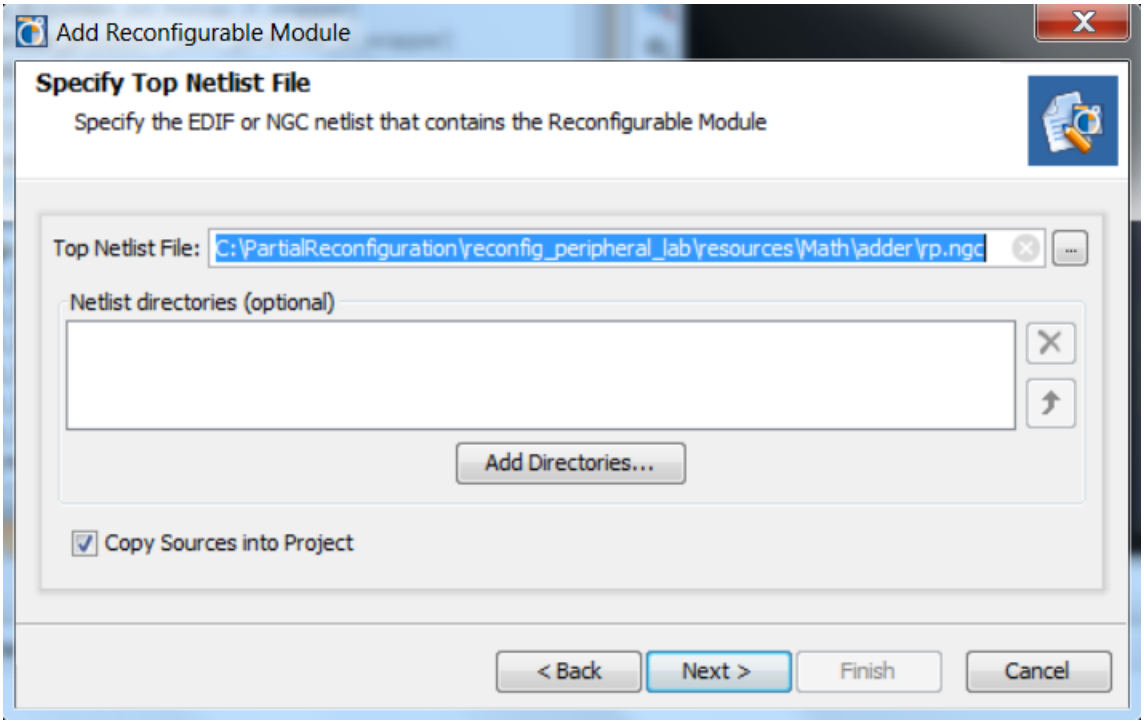

Figure 15: Locating the adder Version of math.ngc

8. Click **Next** *twice*.
9. Click **Finish**.

   In the Netlist pane, expand Reconfigurable Modules hierarchy under math_0/USER_LOGIC_I/rp_instance to view the adder RM entry.
10. Follow the steps in Step 5 to add a **multiplier** RM from the `<Extract_Dir>/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.

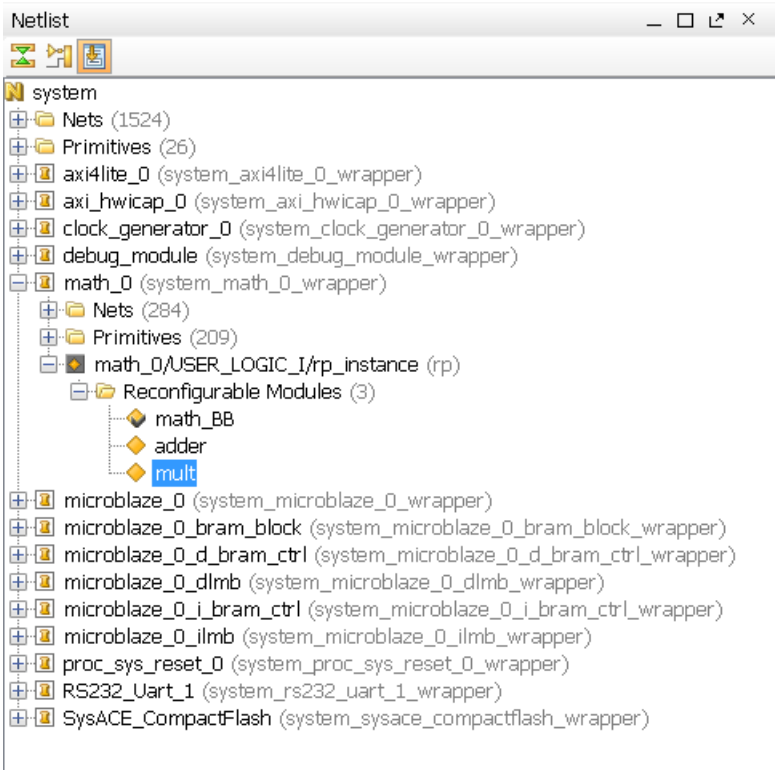

Figure 16: PlanAhead Project with adder and mult RMs added

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. Zoom to the top left quarter of the FPGA.
2. Select **Window > Physical Constraints**.
3. In the Physical Constraints tab, select `pblock_math_0/USER_LOGIC_I/rp_instance`.
4. Right-click, and select **Set Pblock Size**.

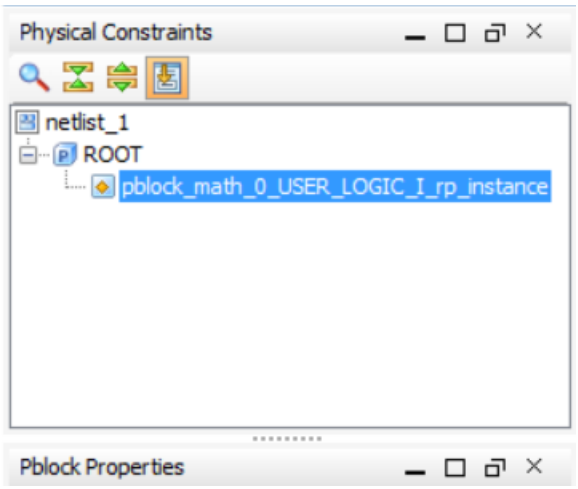

Figure 17: Setting Physical Constraints

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 in the next figure.

   You must draw a box around this region 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**.

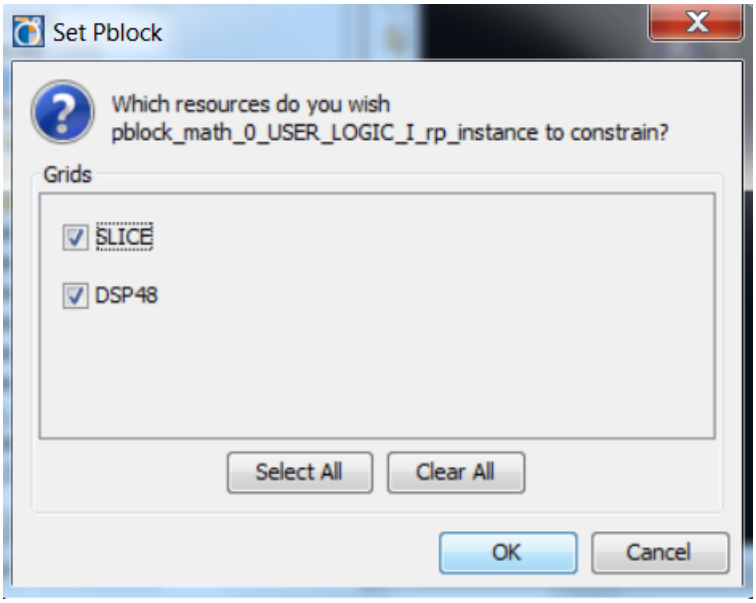

Figure 19: Setting Pblock with SLICE and DSP48

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

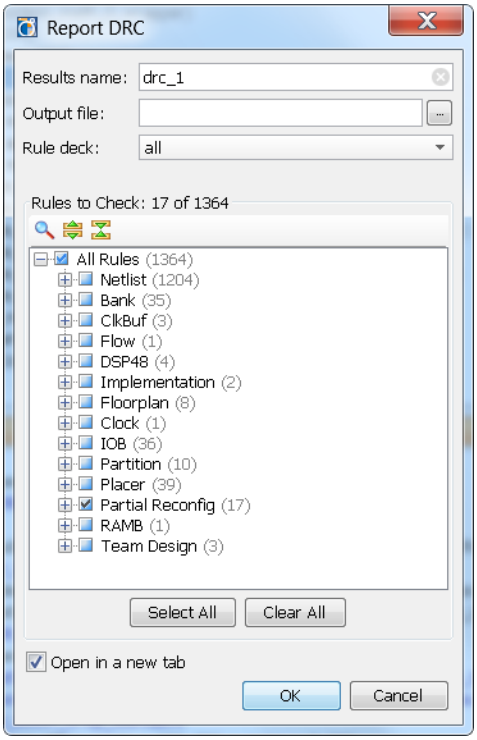

Figure 20: Running Design Rule Checks

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 14** in the Flow drop-down box.
4. Under PlanAhead Strategies, select **ISE Defaults**.
5. Click the **+** button to create a new strategy.

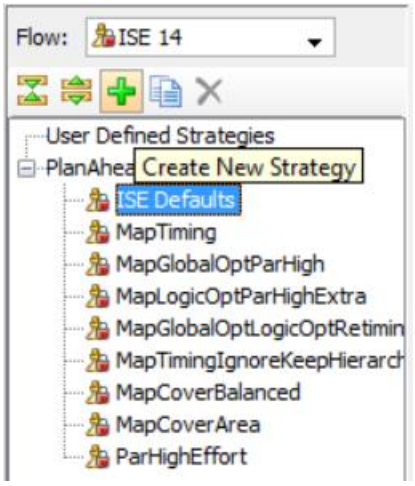

Figure 21: Creating a New Strategy

6. Name the new strategy **ISE14_BM**, and set the Type to **Implement**.
7. Click **OK**.
8. Under Translate (ngdbuild), click in the More Options field.
9. Type **-bm ../../../edk/implementation/system.bmm**, and click **Apply**.
10. Click **OK**.

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**.

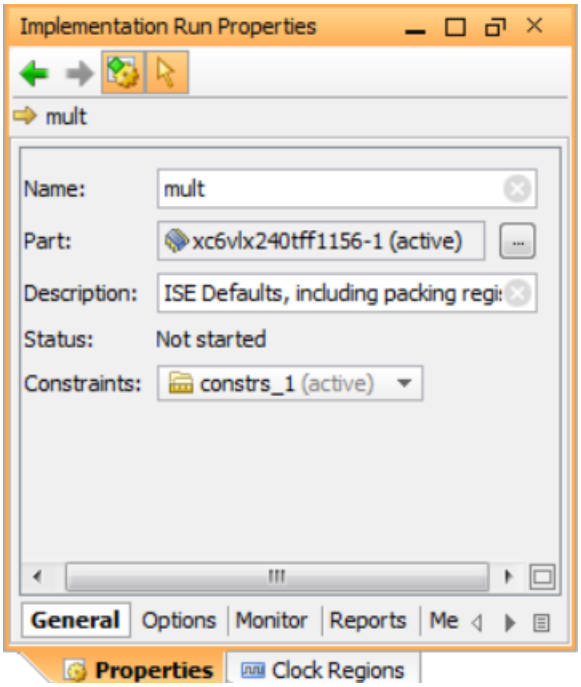

Figure 22: Implementation Run Properties View

6. In the Options tab, change the Strategy to **ISE14_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**.

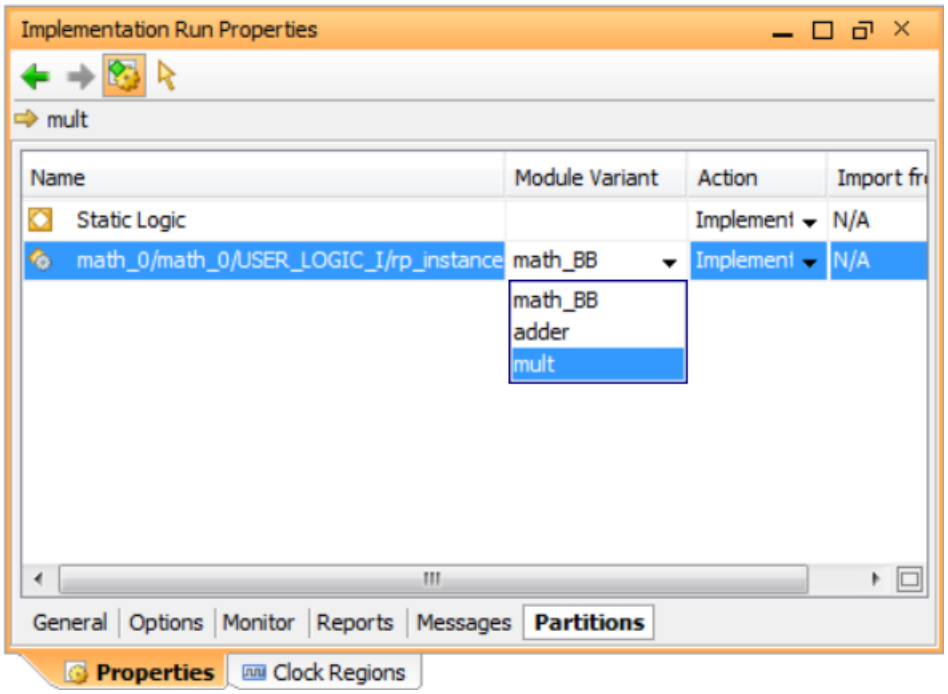

Figure 23: Implementation Run

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** (if a dialog box shows up) to save the project and run the implementation.

    The implementation runs.

    When implementation is finished running, the Implementation Completed dialog box opens to let you open the implemented design, or promote the implemented partitions.
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.

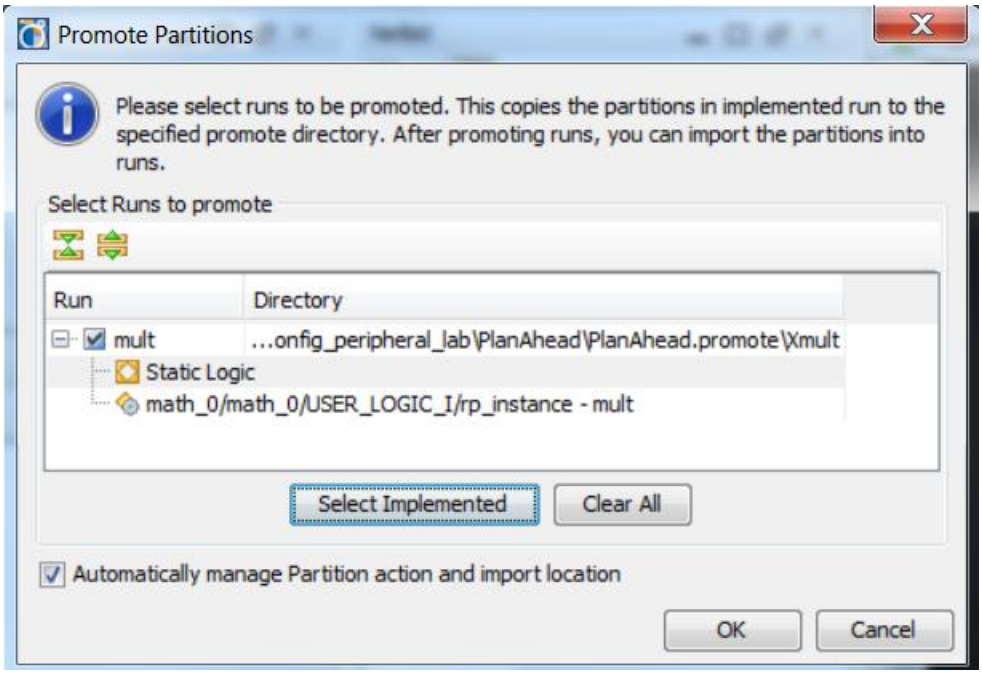

Figure 24: Promoting Partitions

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 Runs**.

   The Create New 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**.

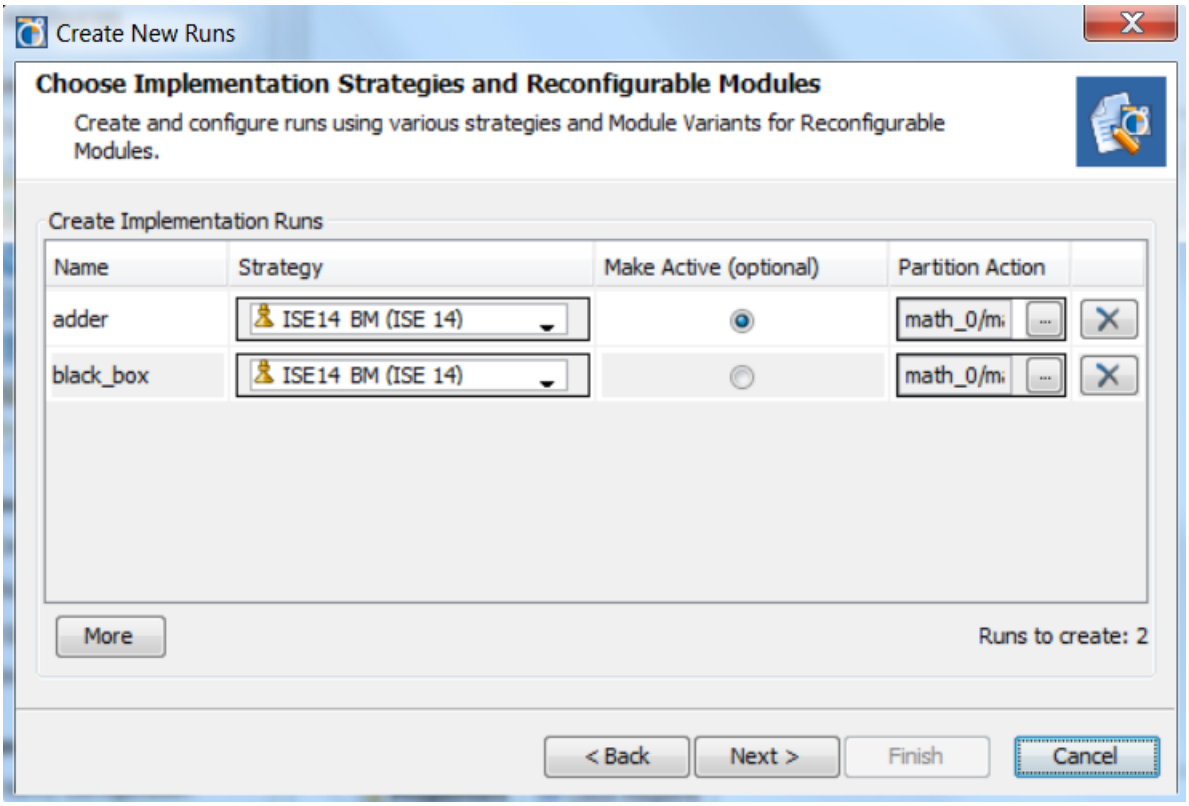

Figure 25: Creating Multiple Runs

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.

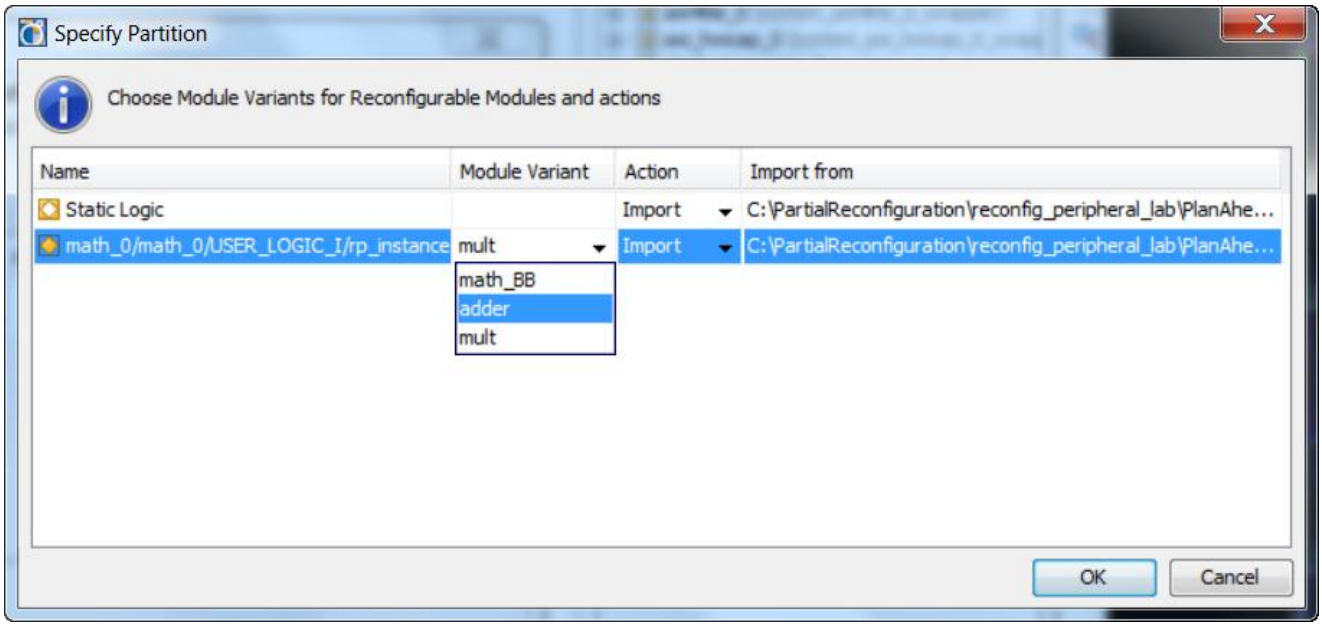

Figure 26: Selecting Adder Module Variant

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

1. Run the PR_Verify utility to make sure that there are no errors.
2. In the Configurations window, select any of the configurations.
3. Right-click, and select **Verify Configuration**.

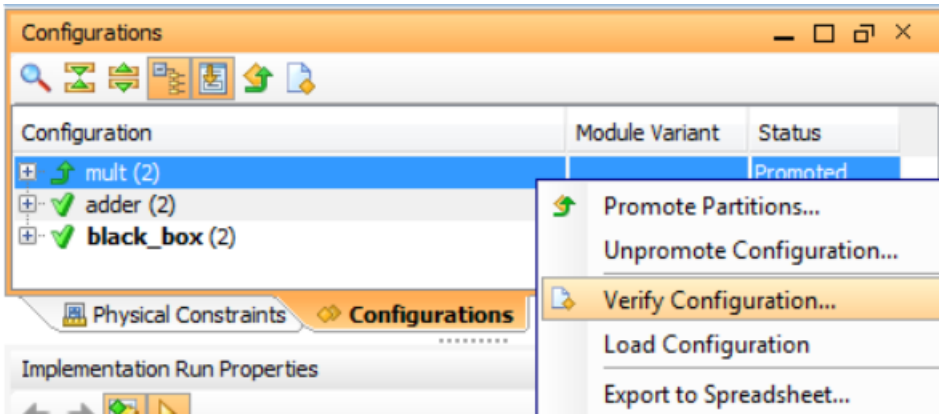

Figure 27: Verifying All Configurations

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

Step 11: Generating Bit Files

After PR_Verify validates all the configurations, 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 `<Extract_Dir>/PlanAhead/PlanAhead.runs/` directory.
3. Click **OK**.
4. Save the project.
5. Close PlanAhead by selecting **File > Exit**.

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 ISE Design Suite command prompt from your Windows environment by selecting **Start > All Programs > Xilinx Design Tools > Xilinx ISE Design Suite > Accessories > ISE Design Suite Command Prompt**.
2. In the command window, go to the `<Extract_Dir>/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 the following table.

File Name	Copy to Directory	Rename File To
<Extract_Dir>/PlanAhead/PlanAhead.runs/adder/adder_math_0_math_0_user_logic_i_rp_instance_adder_partial.bit	/image	adder.bit
<Extract_Dir>/PlanAhead/PlanAhead.runs/mult/mult_math_0_math_0_user_logic_i_rp_instance_mult_partial.bit	/image	mult.bit
<Extract_Dir>/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 `<Extract_Dir>/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.

   Typing O at the menu will let you enter two operands and display the results. Typing M, B, or A at the menu will let you partially reconfigure the multiplication, blanking, addition functionality respectively.

   Try various reconfigurations and enter operands after each reconfiguration to verify that the design indeed works.

   When blanking bitstream is loaded, you should see the output of -1 as there is no logic present in the reconfigurable partition to perform the operation.

Conclusion

In this tutorial, you:

- Created a processor system using XPS.
- Added a user peripheral which included a place holder for the reconfigurable partition.
- Generated netlist files.
- Created an application using SDK.
- Generated a full bitstream as well as partial reconfiguration bitstreams using the PlanAhead tool.
- Generated an ACE file for Compact Flash memory card.
- Verified the functionality using the ML605 evaluation board.