XAPP1162 (v1.0) January 10, 2013

Vivado IP Integrator Step-by-Step

Author: Dinesh Kumar, Yashovardhan Bhatt

Summary

This application note demonstrates the creation of a basic MicroBlaze™ processor based system for a Kintex™-7 FPGA made using Vivado™ IP integrator. The system can be used to run any operating system. This application note demonstrates porting a free RTOS operating system. This system includes native Xilinx® IPs such as Timer, MicroBlaze processor, AXI block RAM, double data rate (DDR), UARTLite, MicroBlaze Debug Module (MDM), proc_sys_reset, and local memory bus (LMB). These are the basic building blocks of any system.

In addition to creating the system described above, this application note also discusses porting an operating system on a Kintex device that uses Xilinx Software Development Kit (SDK) in the Vivado design suite. It prints "hello world" in an OS thread.

The reference system is targeted for the Xilinx KC705 FPGA evaluation board.

Introduction

This design was created using the 2012.4 version of Vivado Design Suite. Versions prior to 2012.4 require a license to use IP integrator.

To test your system on a KC705 board, you must use a terminal emulation program such as TeraTerm/hyperterminal. You must also ensure that the device drivers for the board are correctly installed.

Figure 1 shows the layout of a KC705 board with a Kintex-7 FPGA.

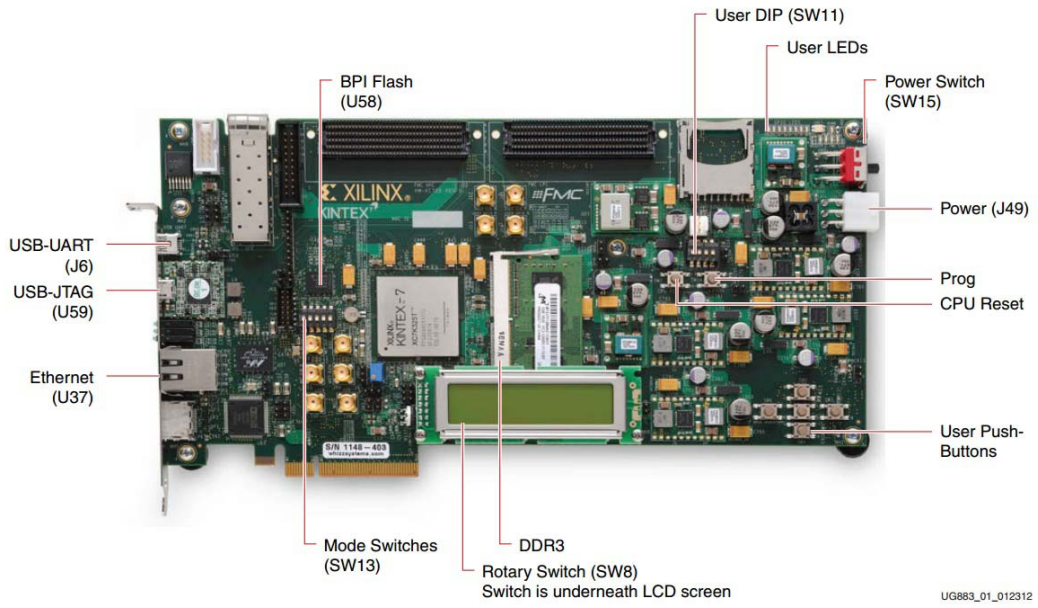

Figure 1: KC705 Board Layout

Table 1: Reference System Address Map

Peripheral	Version	Instance	Base Address	High Address
MicroBlaze	8.40.b	microblaze_1	N/A	N/A
Axi_timer	1.03.a	axi_timer_1	0x41c00000	0x41c0ffff
Axi_uartlite	1.02.a	axi_uartlite_1	0x40600000	0x4060ffff
Axi_INTC	2.00.a	axi_intc_1	0x41200000	0x4120ffff
LMB_BRAM_controller	1.00	mb_bram_if_cntr_1	0x00000000	0x00001fff
mig_7_series	1.7	mig_7_series_1	0x80000000	0xffffffff
AXI4 Interface BRAM controller	2.00.a	axi_vdma_0	0xc0000000	0xc0000fff

Requirements

The following hardware and software are required for this reference system.

Hardware Requirements

- Xilinx Kintex-7 KC705 board
- One USB (Type A to Type B)
- JTAG platform USB cable
- Power cable to the board

Software Requirements

- Vivado Design Suite 2012.4
- SDK14.4

Hardware System Specifics

The hardware design includes:

- MicroBlaze processor
- MicroBlaze Debug Module (MDM)
- Local Memory Bus (LMB) block RAM
- AXI4_INTERCONNECT
- Mig DDR3 RAM
- AXI_BRAM_Interface_controller
- PROC_SYS_RESET
- AXI_TIMER
- AXI_UARTLITE
- AXI_INTC

The MicroBlaze processor acts as controller for other devices and runs the operating system (OS) to prioritize and schedule various tasks. This is a basic system that can be used to build a more complex system on top of this. The IP added in this system are such that they are important for any embedded application and will go a long way in helping you focus only on your IP; you need not worry about the connection or interface of these low-level IP.

AXI4 Interconnect

The design documented in this application note contains two AXI4 Interconnects. One is used for accessing memory through the cache, and the other is used for accessing peripherals. For more information, see the *LogiCORE IP AXI Interconnect Data Sheet* [Ref 1].

- The AXI_Interconnect_1 is used as a bridge between the MicroBlaze processor Instruction cache and Data cache to write into the AXI4 BRAM and the DDR3SDRAM. This Interconnect makes the ICache and DCache buses the master to write into a connected slave memory.
- The AXI_interconnect_2 is used to connect the axi_timer, axi_uartlite, and axi_intc to the MicroBlaze processor AXI Data peripheral bus.

MIG 7 Series

The Memory Interconnect Generator (MIG) is designed to generate DDR3SDRAM to be used in the system. This includes setting the clock frequency, period and type, the data width used, and the DDR pin constraints. The mig_7_series, once re-configured as DDR3 SDRAM, drives the clock in this system and is the slave of the Instruction cache and Data cache buses of the MicroBlaze processor. The 114 pins of the DDR interface are mapped to a interface port and the local constraints of these pins are specified manually by importing a user defined constraint file.

Hardware Reference System

A working system is provided with this document. All necessary external configuration files can be found in this system. The path of these files, wherever used in this document, is specified. Any reference to the path of this system is referred to as `<reference design path>`.

The reference design has been fully verified and tested on the Kintex-7 FPGA KC705 board. The reference design files for this application note are provided in `xapp1162.zip`, which is available as an addition to this document. You can download it from https://secure.xilinx.com/webreg/clickthrough.do?cid=201113.

The reference design checklist is shown in Table 2.

Table 2: Reference Design Checklist

Parameter	Description
General	
Developer name	Dinesh Kumar, Yashovardhan Bhatt
Target devices (stepping level, ES, production, speed grades)	Kintex 7 (KC705) FPGA
Source code provided	Yes
Source code format	VHDL/Verilog (some sources encrypted)
Implementation	
Synthesis software tools/version used	Vivado Design Suite 2012.4
Implementation software tools/versions used	Vivado Design Suite 2012.4
Hardware Verification	
Hardware verified	Y
Hardware platform used for verification	KC705 board

Creating a New Project

1. Open the Vivado integrated design environment (IDE).
2. In the Welcome window, click **Create New Project**.

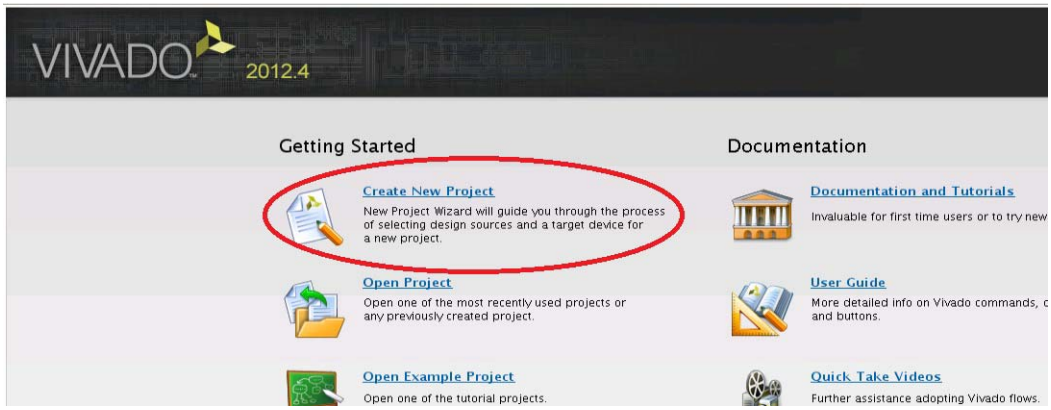

Figure 2: Creating New Project

The New Project wizard opens.

3. Click **Next**.
4. Type the project name and location.

   Caution! Do not use spaces in the project name or location name.
5. Click **Next**.

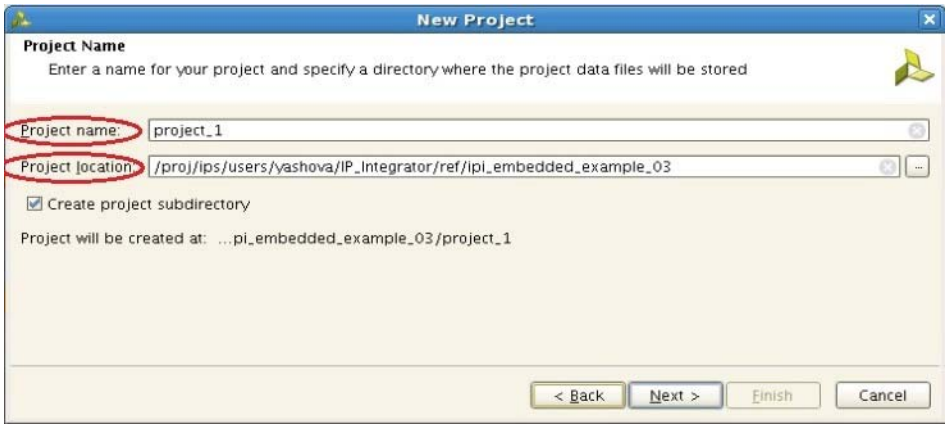

Figure 3: Naming Your Project

6. Select the **Create Project Subdirectory** check box.
7. Click **Next**.

   For this project, do not add any sources.
8. Select your target language. For this project, select Verilog.
9. Click **Next**.

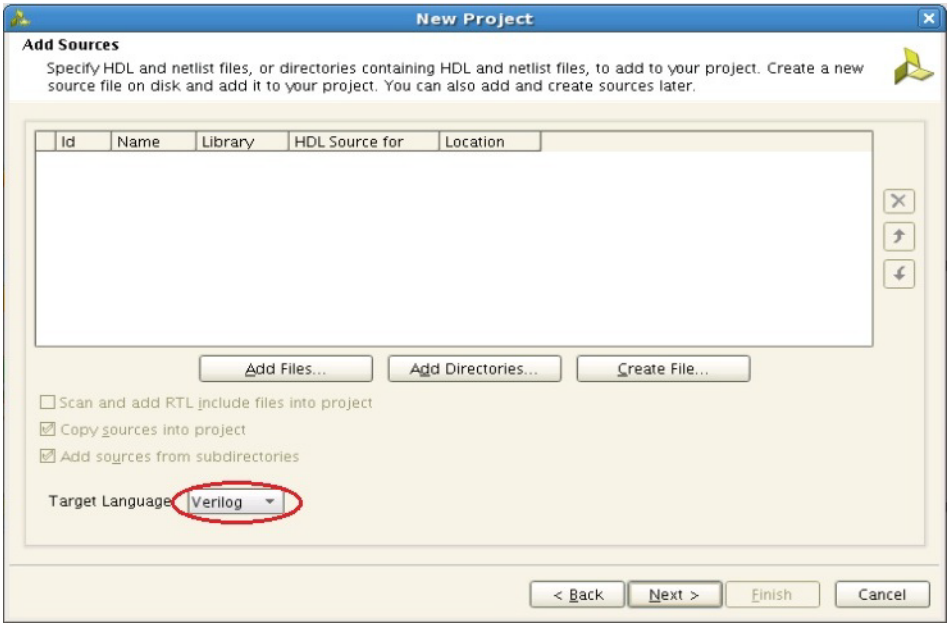

Figure 4: Selecting Verilog as Target Language

For this project, do not add IPs.

10. Click **Next**.

    For this project, do not add constraints.

11. Click **Next**.
12. Click **Boards**.
13. Select the Kintex-7 KC705 Evaluation Platform.
14. Click **Next**.

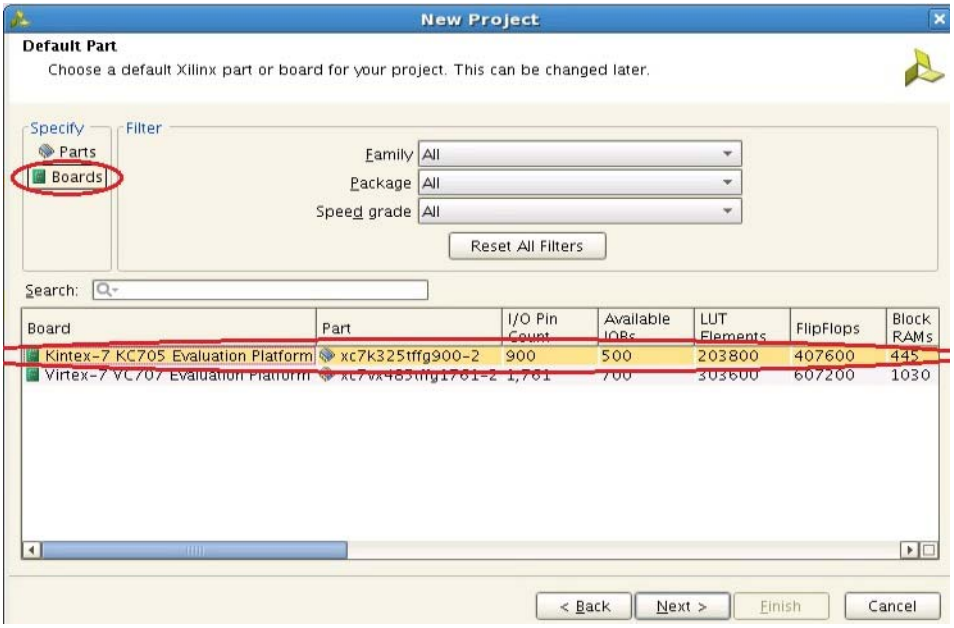

Figure 5: Selecting Kintex-7 KC705 Board

15. Click **Next**.

    The project summary displays.

16. Click **Finish** to create your project.

Creating a New Block Diagram

In this section, you will create a new project.

1. Click the **Create Block Design** option in the IP Integrator tab of the Flow Navigator.
2. Type a name for the design.
3. Click **OK**.

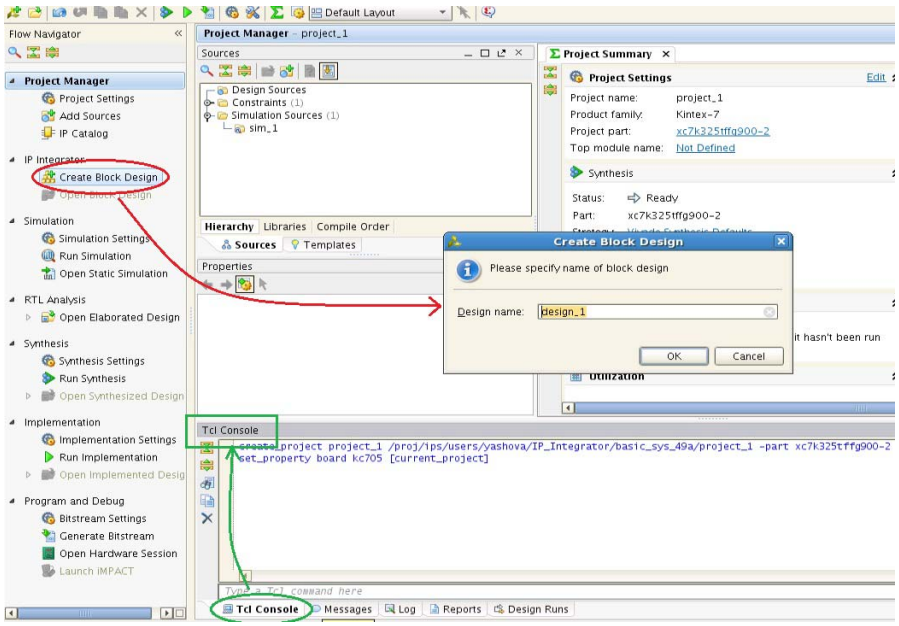

Figure 6: Creating a New Project

The Tcl Console displays the equivalent Tcl commands for each action.

Adding IP

First, add the MicroBlaze processor IP.

1. Right-click in the block diagram window and select **Add IP**.
2. In the IP catalog that opens, type **`microblaze`** in the search bar and press the **Enter** key. You can also add multiple IPs by dragging them from the catalog to the block diagram.

The IP is added to the diagram. The IP integrator allows you to change the position of an IP in the diagram: horizontally in rigid vertical columns and vertically in a column.

Next, add the following IPs, shown in Figure 7:

- MIG 7 Series Memory Interface Generator
- MicroBlaze Debug Module
- Proc Sys Reset

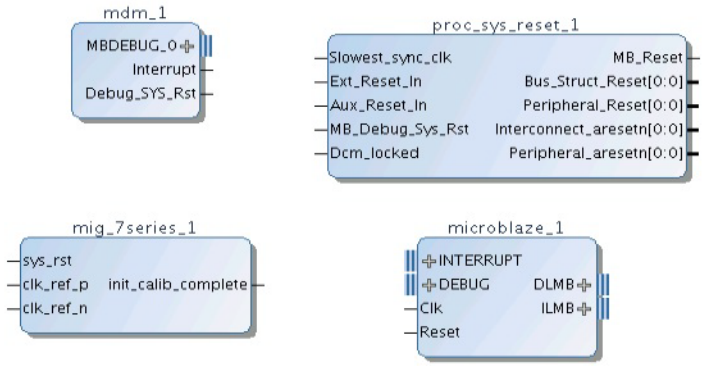

Figure 7: Add MicroBlaze processor, MDM, MIG 7 Series, and Proc System Reset IPs

Navigating the Block Diagram

The main window of IP integrator has a toolbar on the left. Figure 8 displays descriptions of the toolbar buttons.

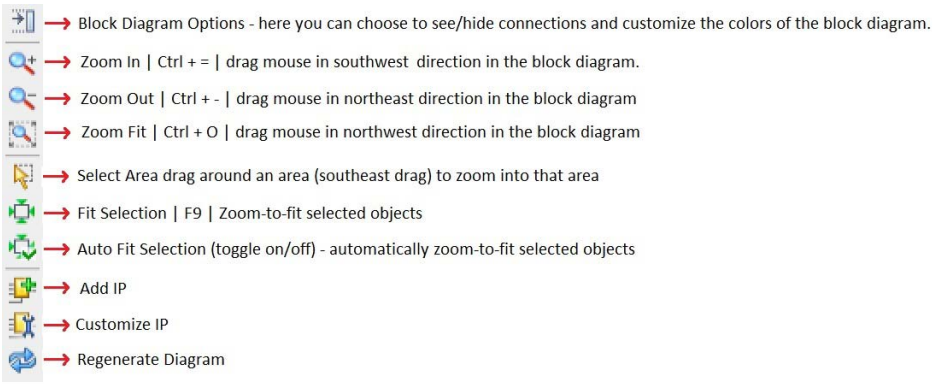

Figure 8: IP Integrator Toolbar Description

Customizing DDR

Do the following to use the Xilinx Memory Interface Generator to customize the double data rate (DDR):

1. Double-click the mig_7_series to customize the mig_7_series (DDR3) IP.

   The Memory Interface Generator opens.

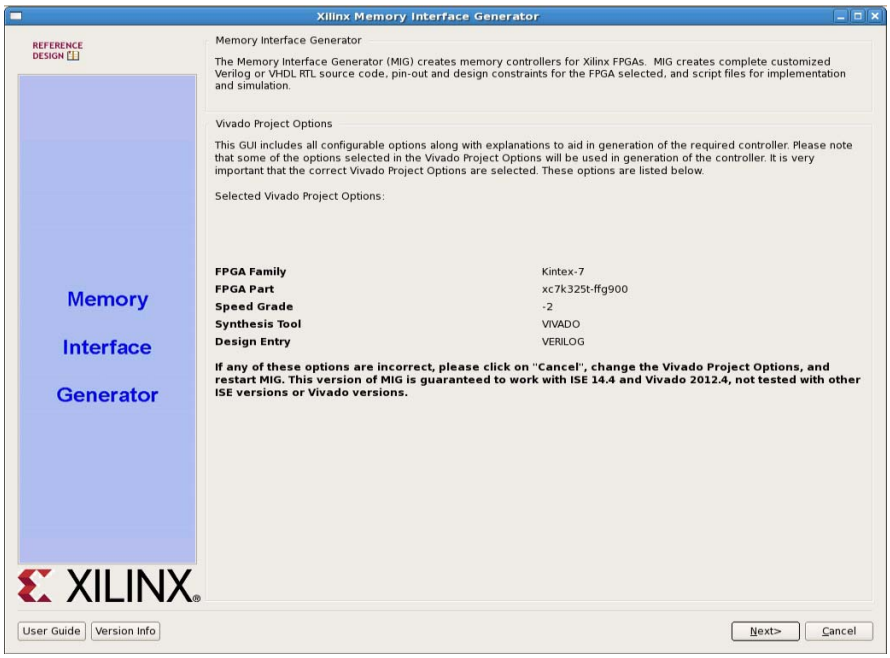

Figure 9: Memory Interface Generator

2. Click **Next**.

   By default, the create design option is enabled and the number of controllers is set to 1.

3. Select the **AXI4 Interface** check box to enable the AXI4 Interface.

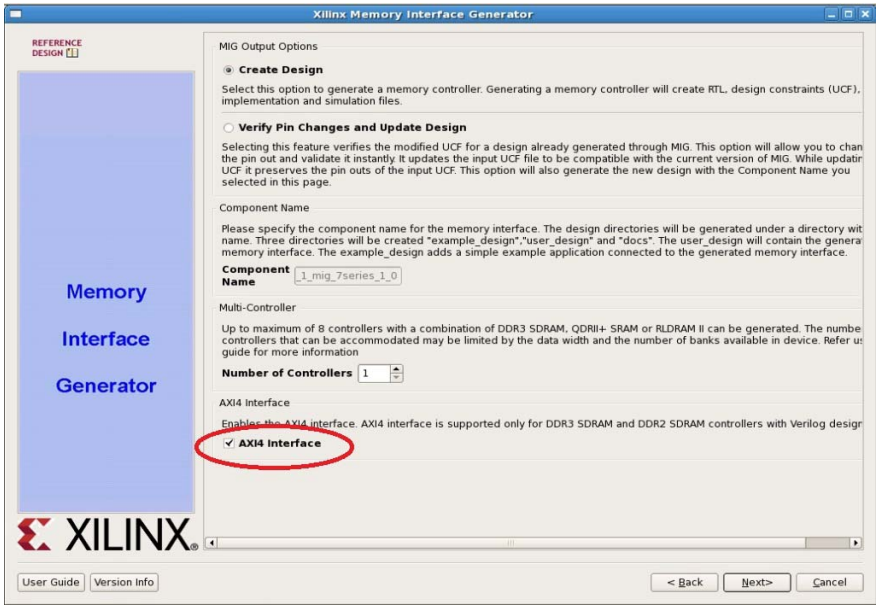

Figure 10: Enable the AXI4 Interface

4. Click **Next**.

   Do not select pin compatible FPGAs.

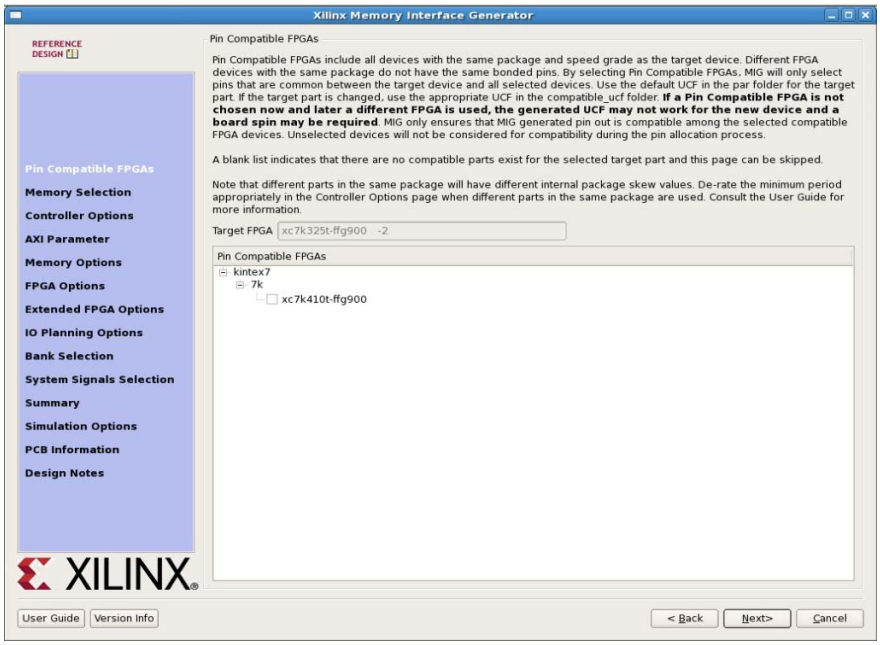

Figure 11: Pin Compatible FPGA Options

5. Click **Next**.

   DDR3 SDRAM is selected by default.

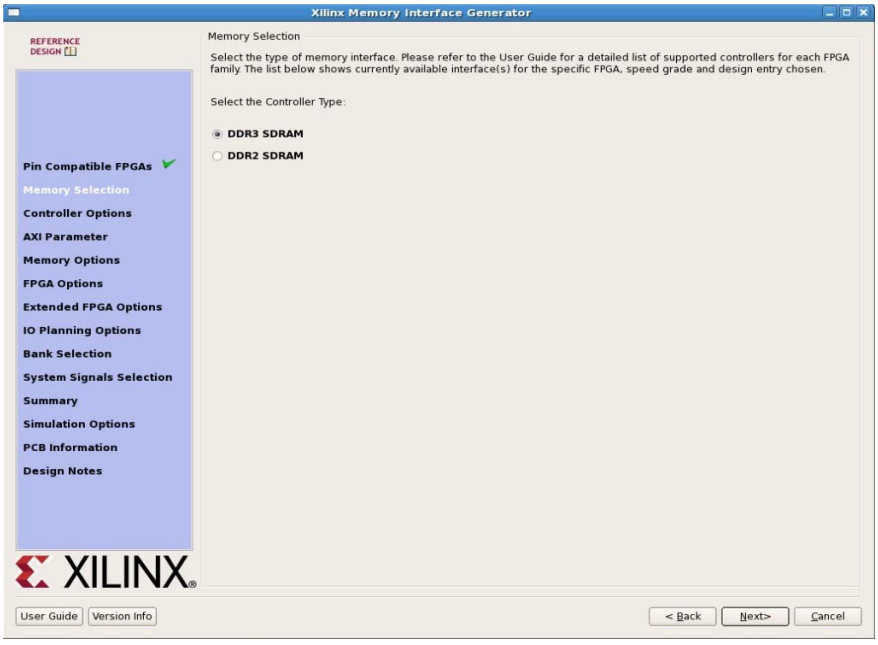

Figure 12: Memory Selection Options

6. Click **Next**.

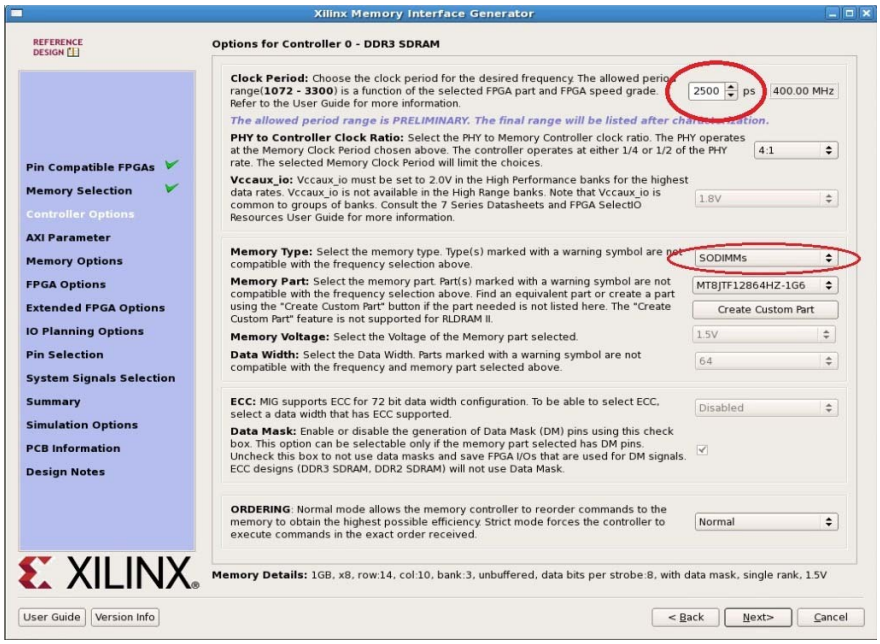

Figure 13: Controller Options

7. Set the value of the clock period to **2500** ps (400.00 MHz)
8. Select the memory type as SODIMMs. This changes the Memory Part to **MT*JTF12864HZ-1G6**.
9. Click **Next**.

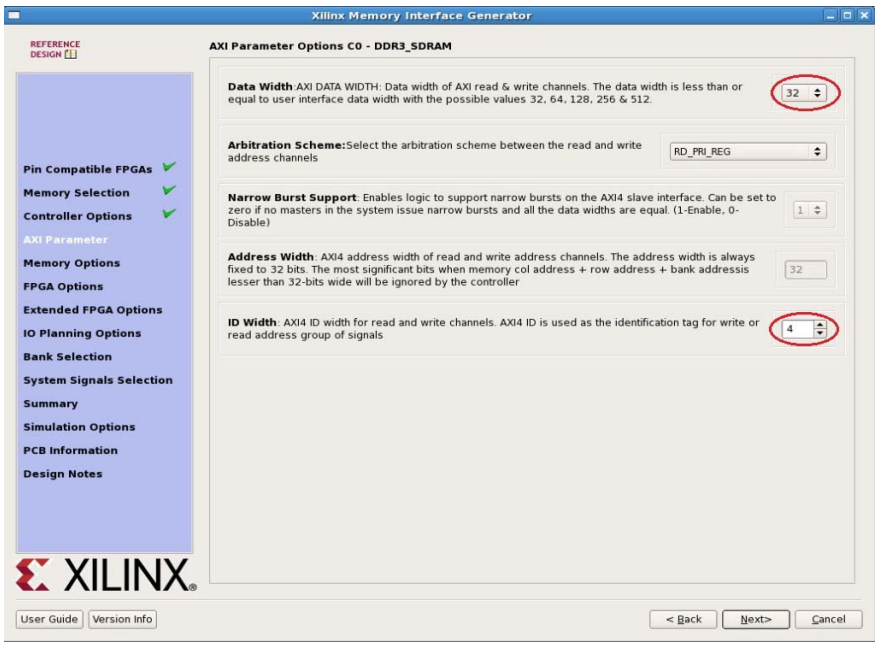

Figure 14: AXI Parameter Options

10. Set Data Width to **32**.
11. Set Data ID Width to **4**.

12. Click **Next**.

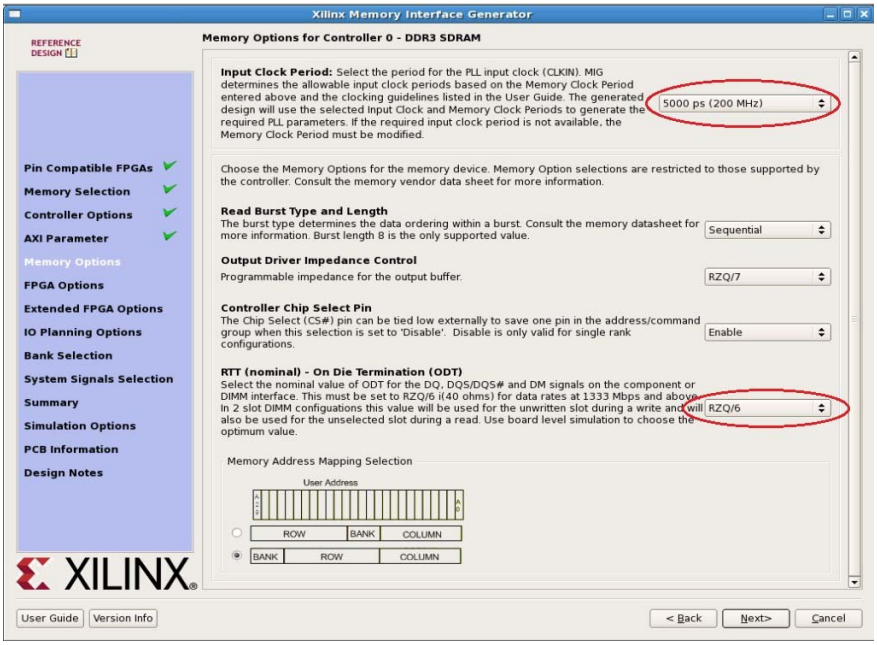

Figure 15: Memory Options

13. Set the Input Clock Period to **5000 ps (200 MHz)**.
14. Set nominal value of the On Die Termination to **RZQ/6**.
15. Click **Next**.

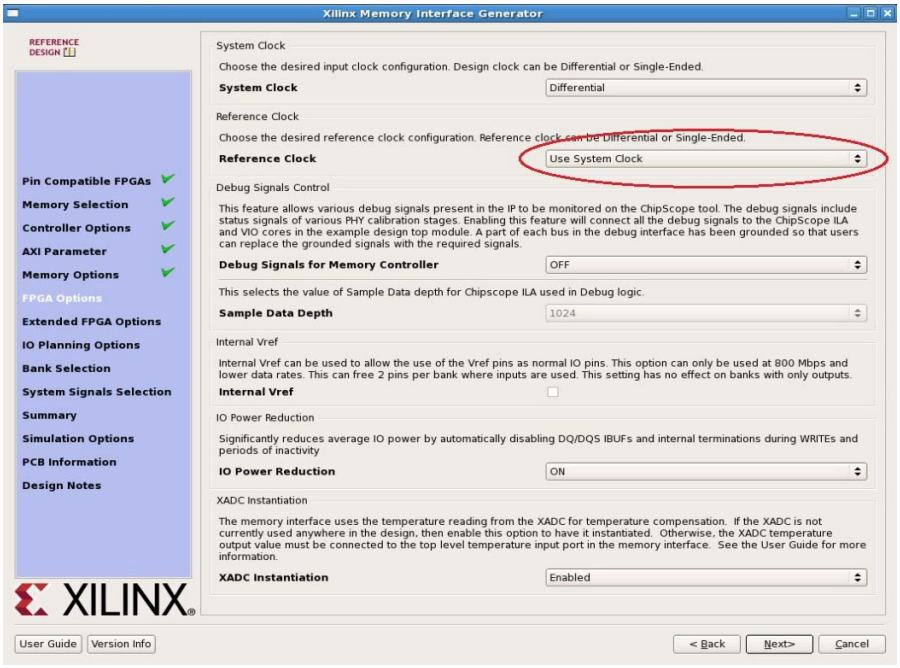

Figure 16: FPGA Options

16. Set the Reference Clock configuration to **Use System Clock**.

17. Click **Next**.

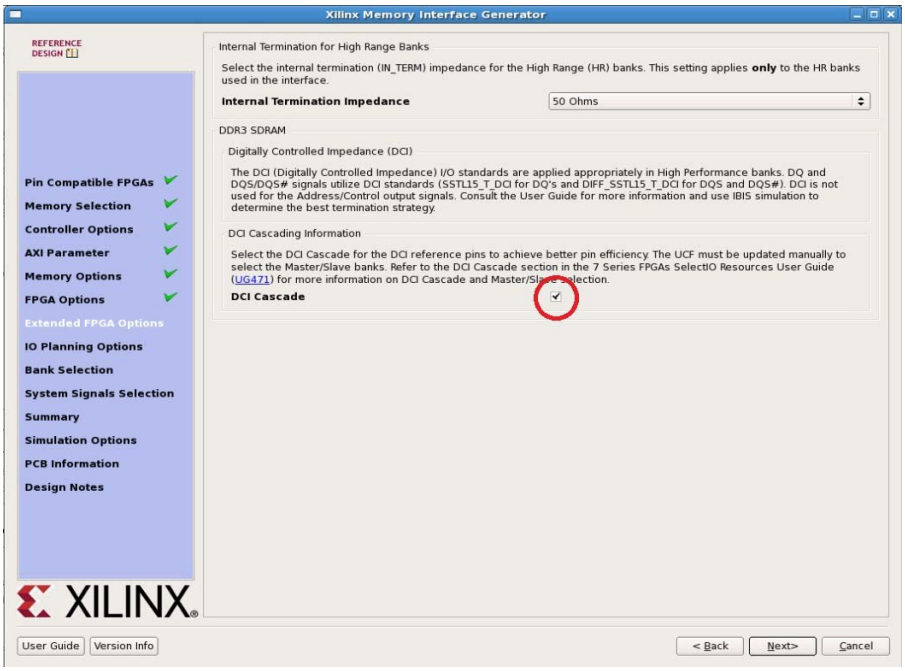

Figure 17: Extended FPGA Options

18. Select the **DCI Cascade** check box.
19. Click **Next**.

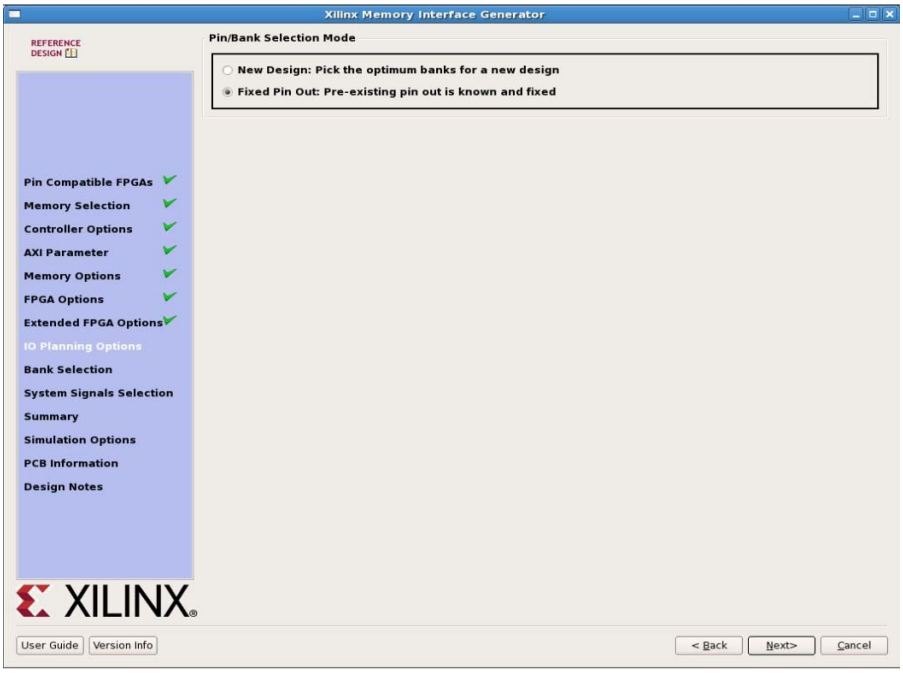

Figure 18: IO Planning Options

20. Select the **Fixed Pin Out** Pin/Bank Selection mode. You will import the pin configurations a specified user constraints file (UCF).

21. Click **Next**.

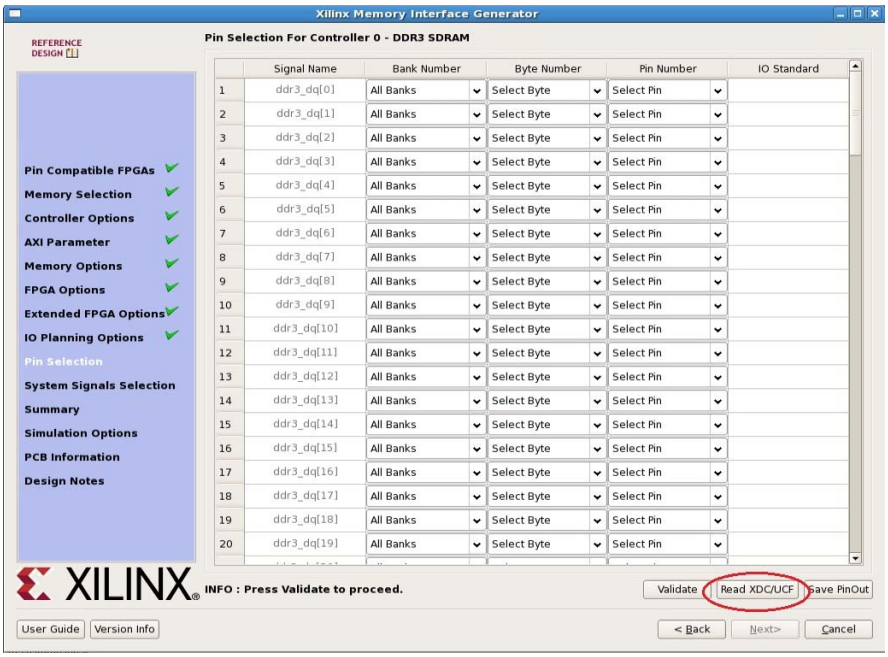

Figure 19: Pin Selection - Import UCF to Configure DDR Pins

22. In the pin selection window, click **Read XDC/UCF**.
23. Select the pin configuration, and click **Open** to import it.

Note: The pin configuration can be found at `<reference design path>/IPI_DDR_XAPP/mig_7_series_pin_layout.ucf`.

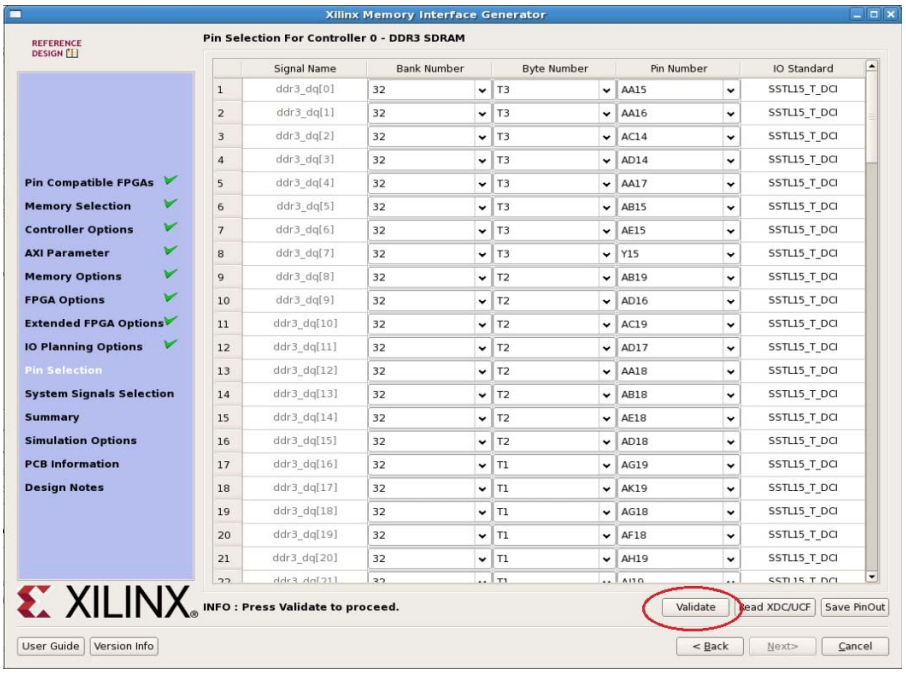

Figure 20: Validate Pin Configuration

24. Click **Validate** to open the validation results log.

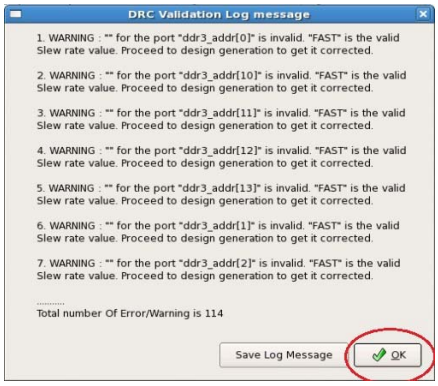

Figure 21: Accept Validation Log Messages

25. After reviewing the log, click **OK**.

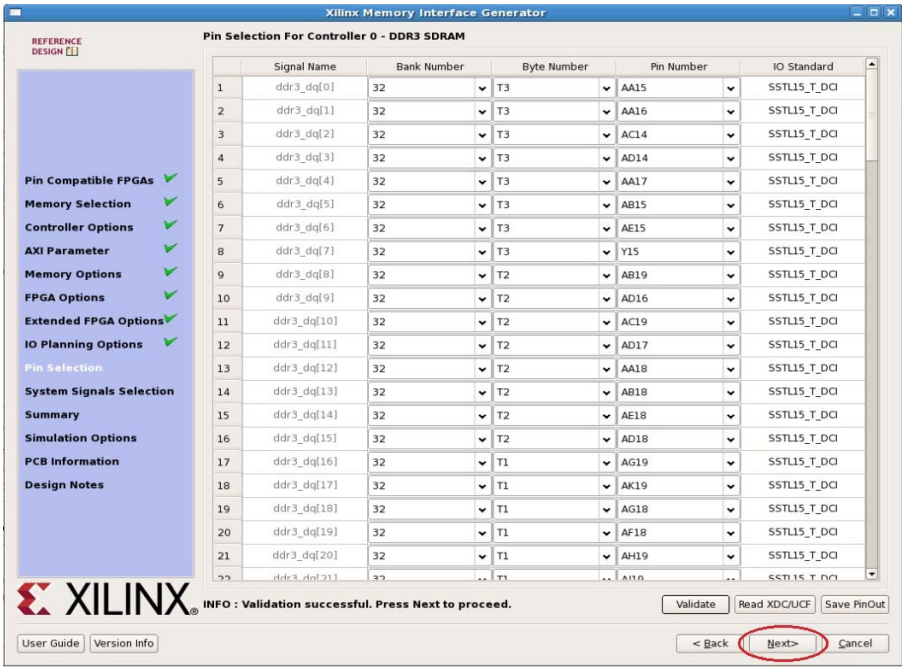

Figure 22: Pin Selection Complete

26. Click **Next** to show the clocking configurations.
27. Click **Next** to open a summarized report.
28. Click **Next** to open the simulation model license agreement.
29. Click **Accept** to accept the license agreement.
30. Click **Next** to open a PCB configuration note.
31. Click **Next**.

    The design notes opens.

32. Click **Generate** to generate the mig_7_series ddr3 IP.

    Figure 23 shows the configuration of Mig_7_series DDR3 IP after customizing it using the Xilinx Memory Interface Generator.

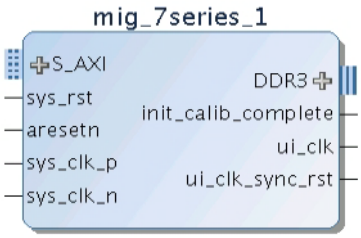

Figure 23: DDR3SDRAM

Customizing a MicroBlaze Processor

1. To customize a MicroBlaze processor, double-click on it.

   The Customization Options dialog box opens.

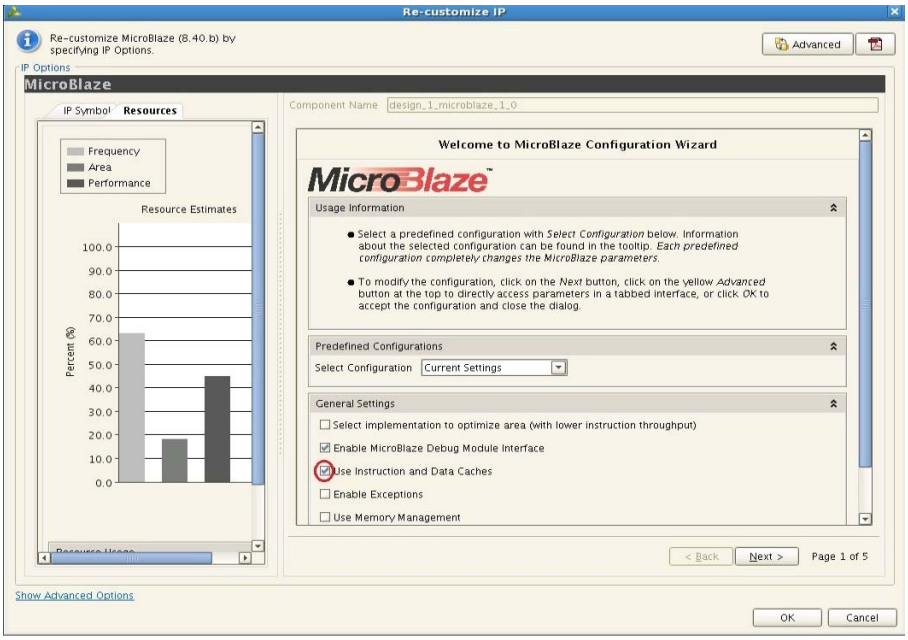

Figure 24: Enable Instruction and Data Caches

2. Enable Instruction and Data caches.

3. Click **Next**.

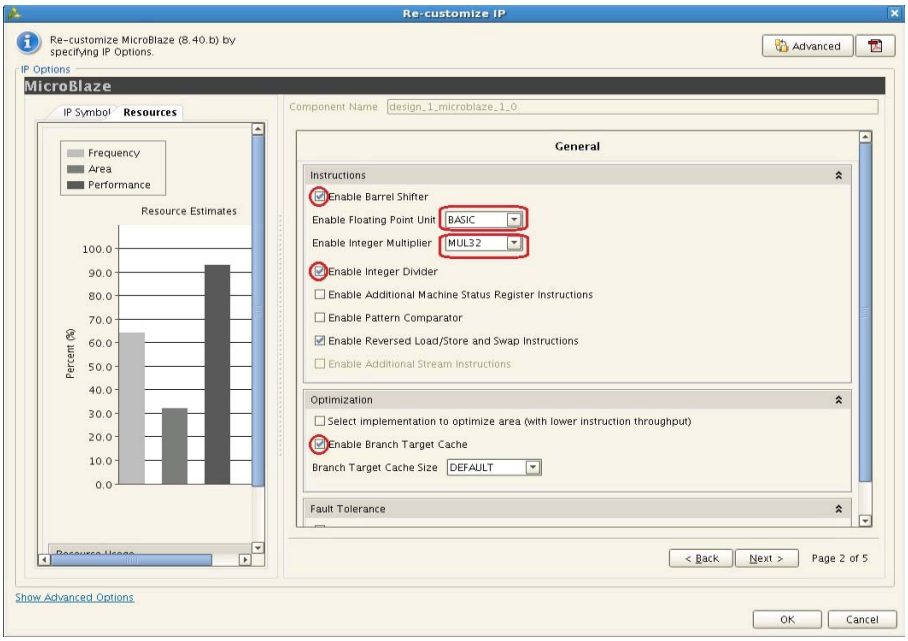

Figure 25: Configure General Settings for MicroBlaze Processor

4. Enable the following:
   - Barrel Shifter
   - Use Integer Multiplier MUL32 (32-bit)
   - Use of basic floating point unit
   - Integer Divider
   - Branch Target Cache
5. Click **Next**.

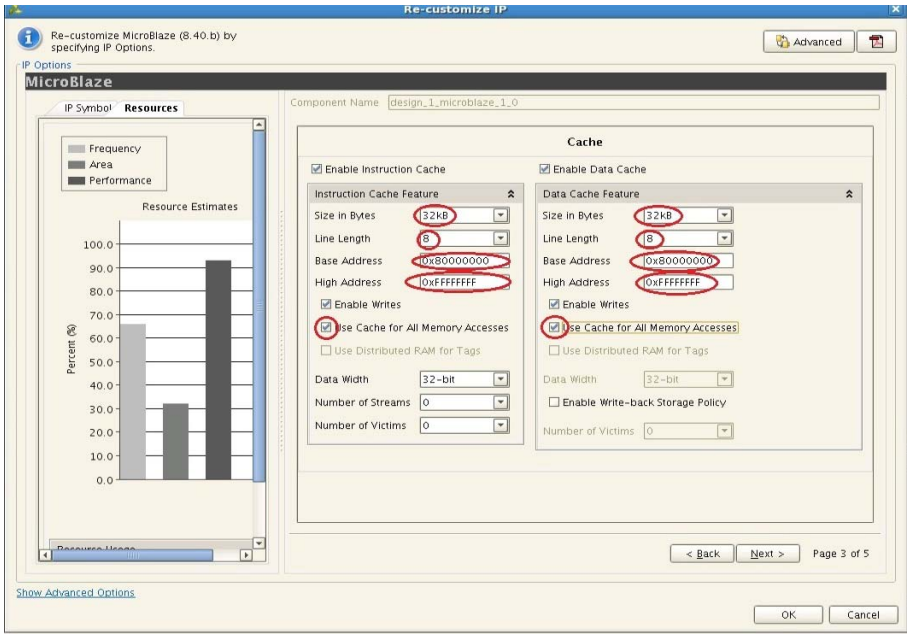

Figure 26: Configure Instruction Cache and Data Cache Settings for MicroBlaze Processor

6. Set the size of both Instruction Cache and Data Cache to **32 KB**.
7. Set line length of both Instruction Cache and Data Cache to **8**.
8. Set High Address of both Instruction Cache and Data Cache to **0xFFFFFFFF**.
9. Set Base Address of both Instruction Cache and Data Cache to **0x80000000**.

   Note: Ensure that you set the High Address before you set the Base Address so that Base Address is not greater than High Address at any time.

10. Enable **Use Cache for All Memory Accesses** for both Instruction Cache and Data Cache.

    Next, ensure that the size of the cacheable segment of memory (that is, the memory space between the Base and High addresses of the Instruction_Cache/Data_Cache) is a power of 2.

    Additionally, the base address and the high address of both Data Cache and Instruction Cache should be the same.

    Ensure that all IPs that are slaves of the ICache and DCache busses fall within this cacheable segment. Otherwise, read/write functions cannot be performed on them.

    Note: For any IP connected only to the Instruction Cache and Data Cache bus, you must enable the **Use Cache for All Memory Access** option. In this example, the ICache and DCache buses are the sole masters of DDR and block RAM; therefore, this option must be enabled. In other configurations, you must decide whether to enable this option as per the design.

11. Click **Next**.

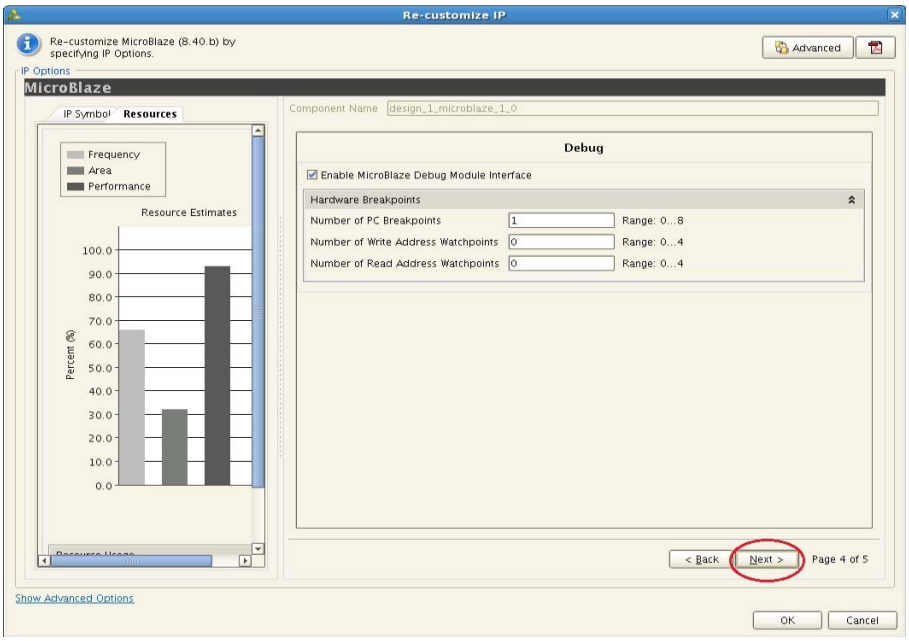

Figure 27: Enable MicroBlaze Debug Module for MicroBlaze Processor

Make sure that MicroBlaze Debug Module is enabled.

12. Click **Next**.

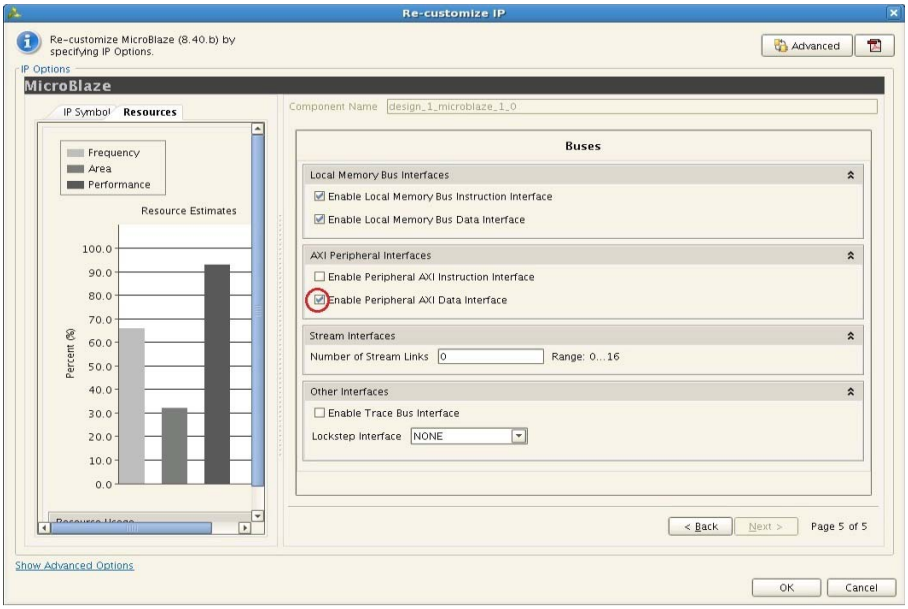

Figure 28: Enable Peripheral AXI Data Interface

13. Check **Enable Peripheral AXI Data Interface**.
14. Click **OK** to generate re-configured MicroBlaze processor.

    Figure 29 shows the configuration of the MicroBlaze processor after re-customizing it.

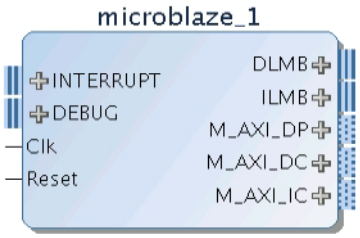

Figure 29: Re-Customized MicroBlaze Processor

Making a Connection Between Two Ports

To make a connection between two ports:

1. Hover your mouse cursor over one of the two ports.

   The cursor changes to a pencil icon to signify that a connection can be "drawn."
2. Click on the port and drag the cursor away from it. After a small distance, you can stop dragging; notice that a connection is following your cursor.

The possible connections are identified with a check mark, as shown in Figure 30.

Note: The ports that are identified are compatible ports; they are not necessarily the correct ports for a connection.

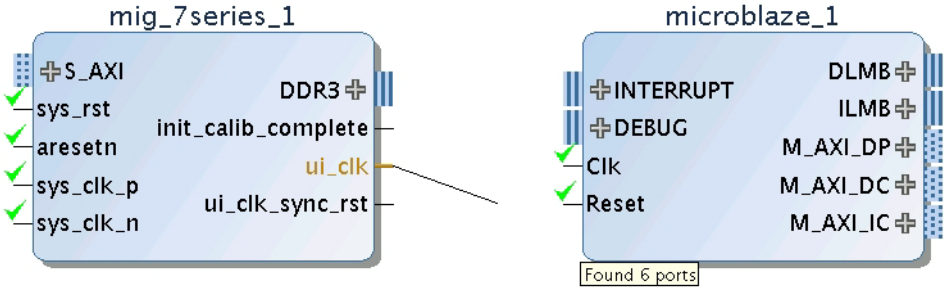

Figure 30: Starting a Connection

3. To complete your connection, click on the connecting port.

   The connection line turns dark to identify that a connection is made.

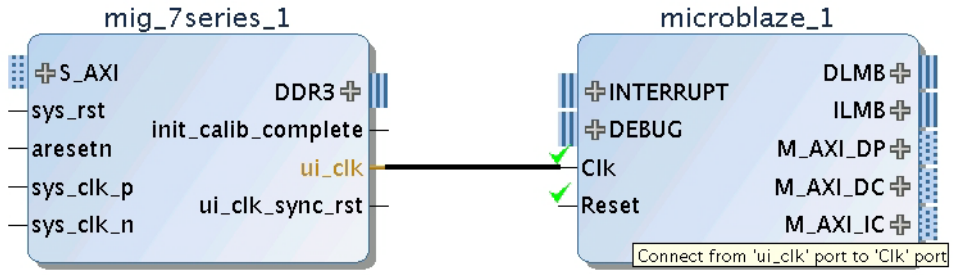

Figure 31: Completing a Connection

Note: If at any time you do not want to make the connection, you can terminate it. To do so, right-click on any empty area in the block diagram and select **End Connection Mode**. Alternatively, you can press the **Esc** key on your keyboard.

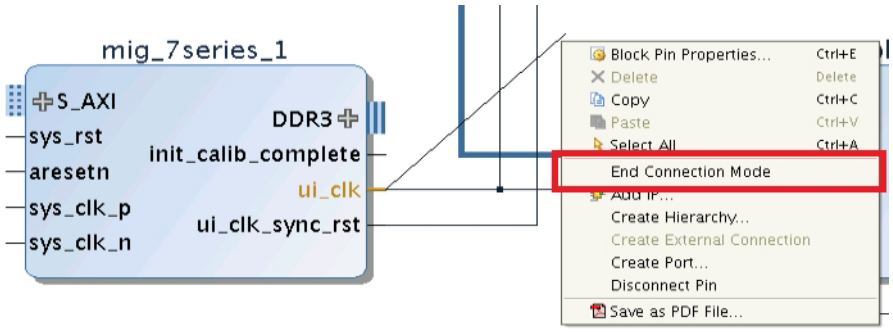

Figure 32: Ending a Connection

Creating First Design Connections

The clock in this system is driven by the `ui_clk` port of mig_7_series DDR3 IP. Make the following connections:

1. Connect the `ui_clk` port of mig_7_series to the `Clk` port of the MicroBlaze processor.
2. Connect the `ui_clk` port of mig_7_series to the `Slowest_sync_clk` port of proc_sys_reset.
3. Connect the `ui_clk_sync_rst` port of mig_7_series to the `Ext_Reset_In` port of proc_sys_reset.
4. Connect the `MB_Reset` port of proc_sys_reset to the `Reset` port of the MicroBlaze processor.
5. Connect the `MBDEBUG_0` bus interface of the mdm to the `DEBUG` interface of MicroBlaze processor.
6. Connect the `Debug_SYS_Rst` port of mdm to the `MB_Debug_Sys_Rst` port of proc_sys_reset.

Figure 33 shows the connected system.

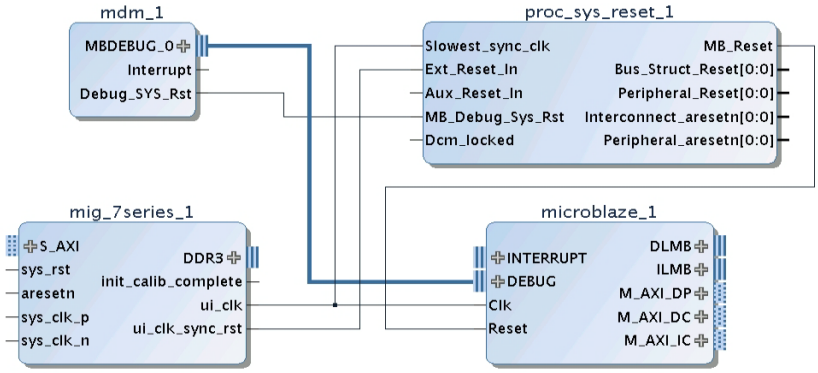

Figure 33: Creating First Connections

Adding Local Memory

1. Add two local memory buses (LMB), two LMB block RAM controllers, and one block memory generator.

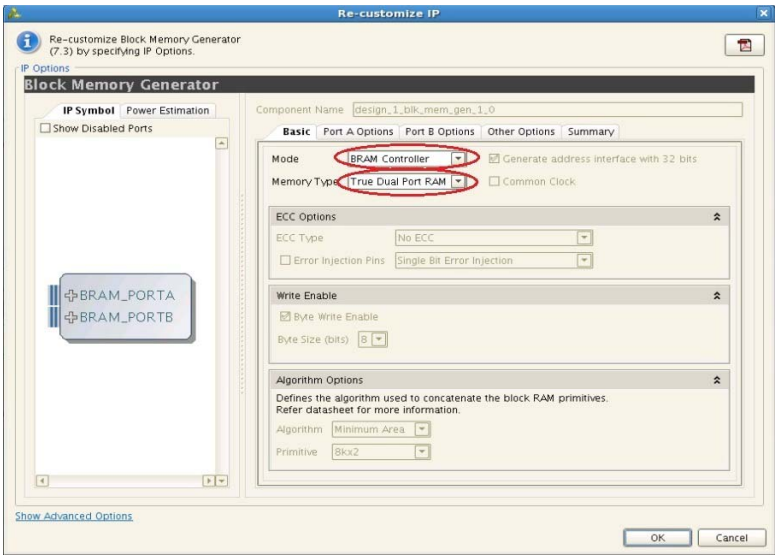

Figure 34: Configuring Block Memory Generator

2. In the Basic tab, customize the **block memory generator** as follows:
   - Set Mode to **BRAM Controller**.
   - Set the Memory Type to **True Dual Port RAM**.
3. Make the following connections. The connected system is shown in Figure 35.
   - Connect the `ui_clk` port of mig_7_series to all `LMB_Clk` ports of the local memory buses and the lmb_bram_controllers.
   - Connect the `Bus_Struct_Reset[0:0]` port of proc_sys_reset to the `SYS_Rst` ports of the local memory buses.
   - Connect the `LMB_M` bus of one LMB to the `DLMB` bus of the MicroBlaze processor, and the `LMB_M` bus of the other LMB to the `ILMB` bus of the MicroBlaze processor.

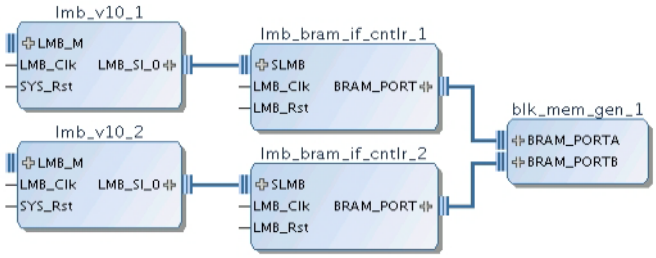

Figure 35: Creating Local Memory

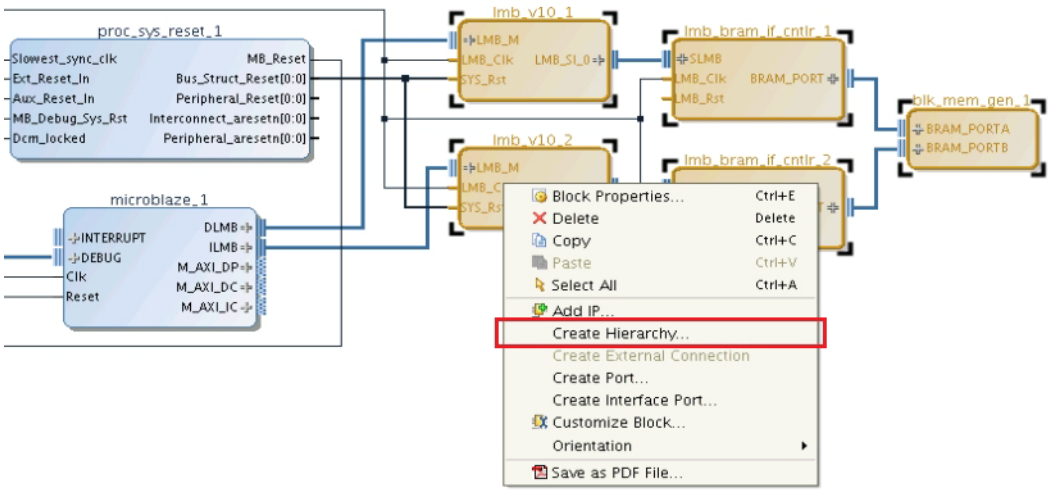

Figure 36: Creating a Hierarchy - Local Memory

4. **Ctrl** + click to select the five IPs: two LMB, two LMB block RAM controllers, and one block memory generator. Right-click and select **Create Hierarchy**. Name the hierarchy.

   The five blocks collapse into a wrapper box, as shown in Figure 37.

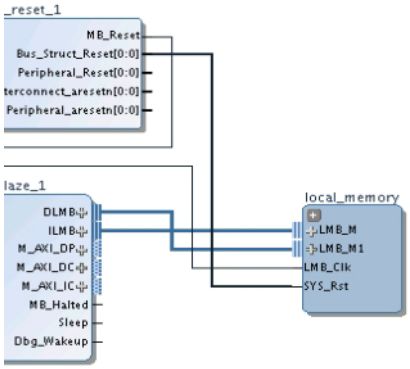

Figure 37: Hierarchy Collapsed

5. Click the **+** sign at the top left corner of the hierarchy box to expand the hierarchy.

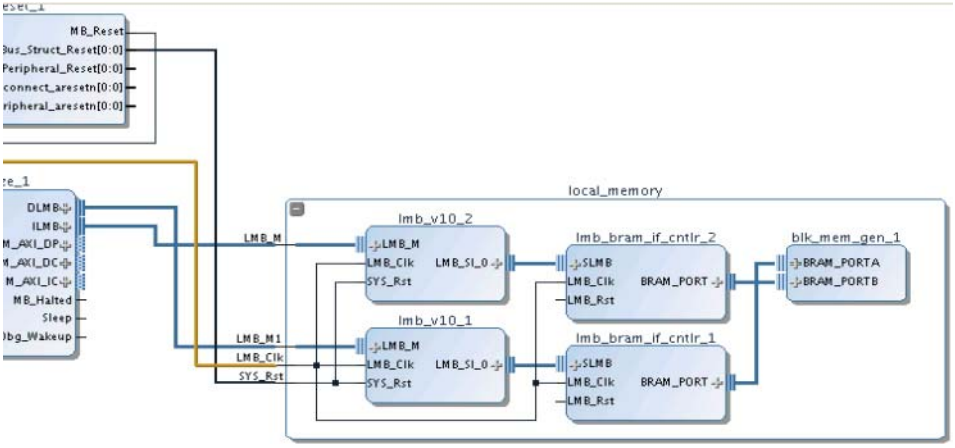

Figure 38: Hierarchy Expanded

Adding Memory

1. Add an AXI4 Interconnect and customize it for two slave ports and two master ports.

   Note: In this design we will be adding AXI4 block RAM Interface controller and mig_7_series to this AXI4 Interconnect. If you need to add more IPs to interconnect, update this accordingly.

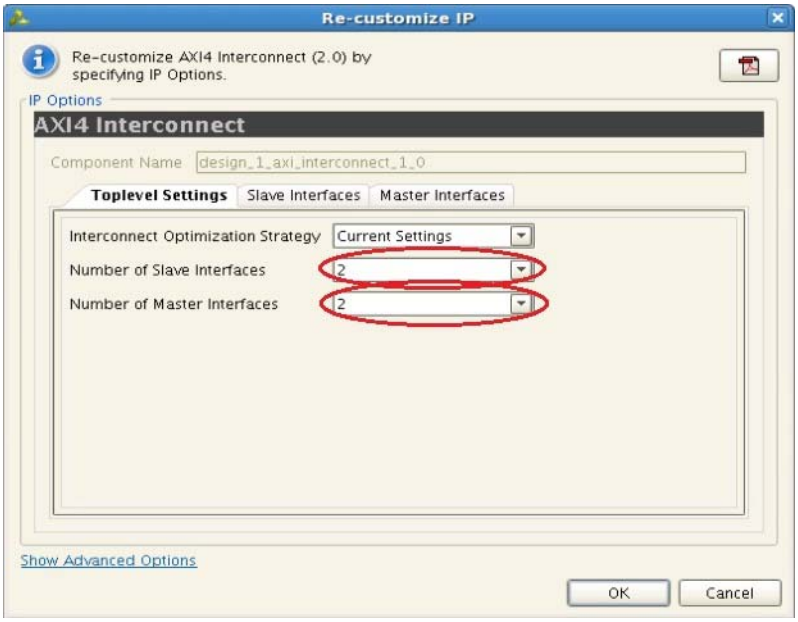

Figure 39: Customizing AXI4 Interconnect

2. To one slave Interface of the AXI4 Interconnect `S00_AXI`, connect the `M_AXI_DC` (DCache) interface of the MicroBlaze processor. To the other slave Interface of the AXI4 Interconnect, such as `S01_AXI`, connect the `M_AXI_IC` (ICache) interface of the MicroBlaze processor.

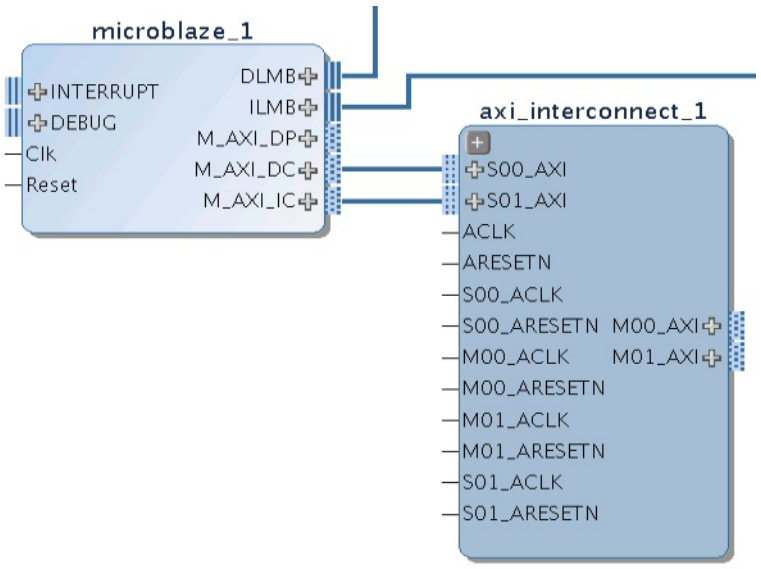

Figure 40: Connecting ICache and DCache to the AXI4 Interconnect

3. Add an AXI4 Interface Bram controller and a block memory generator.

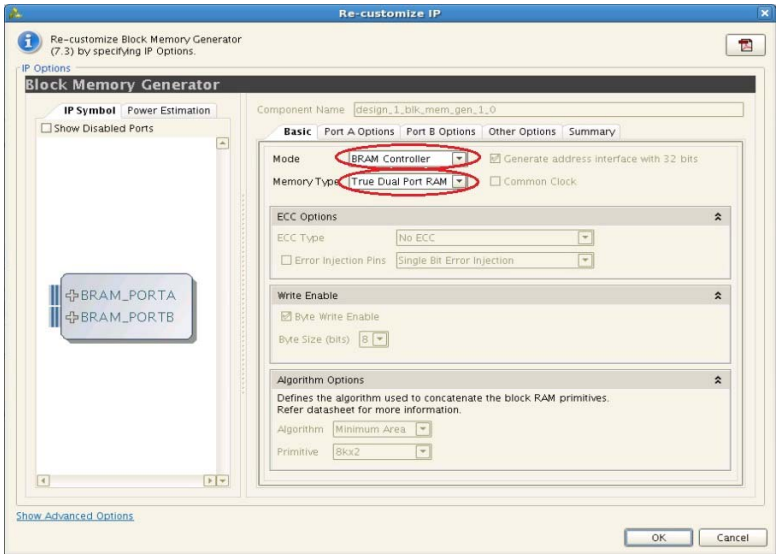

Figure 41: Customizing Block Memory Generator

4. Configure the block memory generator for BRAM Controller mode and True Dual Port RAM memory type.
5. Make the following connections:
   a. Connect `BRAM_PORTA` of the Block Memory Generator to `BRAM_PORTA` of the AXI4 Interface Bram controller, and `BRAM_PORTB` of the Block Memory Generator to `BRAM_PORTB` of the AXI4 Interface Bram controller.
   b. Connect the `S_AXI` bus of the AXI4 Bram controller to the master interface `M00_AXI` of the AXI4 Interconnect (axi_interconnect_1).
   c. Connect the `S_AXI bus` of the mig_7_series DDR3 to the other master port `M01_AXI` of the AXI4 Interconnect (axi_interconnect_1).
   d. Connect all clock ports of the AXI Interconnect (`ACLK`, `Sxx_ACLK`, `Mxx_ACLK`) clock to the `ui_clk` port of mig_7_series DDR3.
   e. Connect the `S_AXI_ACLK` port of the AXI4 Interface Bram controller to the `ui_clk` port of the mig_7_series.
   f. Connect the `ARESETN` port of the AXI4 Interconnect to the `Interconnect_areset[0:0]` port of proc_sys_reset.
   g. Connect the `Sxx_ARESETN` pins and the `Mxx_ARESETN` pins of the AXI4 Interconnect to the `Peripheral_aresetn[0:0]` pins of proc_sys_reset.
   h. Connect the `S_AXI_ARESETN` port of the AXI4 Interface Bram controller and the `aresetn` port of mig_7_series to the `Peripheral_aresetn[0:0]` port of proc_sys_reset.
6. The connections you just made are shown in Table 3. Each row in the table represents a connection/net to be made. Refer to Figure 42 for these connections.

Table 3: **Memory Connections**

	AXI4 Interconnect_1	Mig 7 Series	AXI4 Interface BRAM Controller	Block Memory Generator	Proc_sys_reset
1			BRAM_PORTA	BRAM_PORTA	
2			BRAM_PORTB	BRAM_PORTB	
3	M00_AXI		S_AXI		
4	M01_AXI	S_AXI			
5	ACLK S00_ACLK S01_ACLK M00_ACLK M01_ACLK	ui_clk	S_AXI_ACLK		
6	ARESETN				Interconnect_aresetn[0:0]
7	S00_ARESETN S01_ARESETN M00_ARESETN M01_ARESETN		S_AXI_ ARESETN		Peripheral_aresetn[0:0]

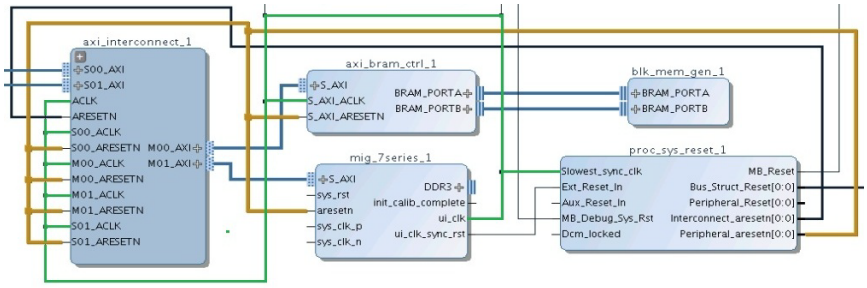

Figure 42: AXI Interconnect to Memory Connections

Adding Peripherals

1. Add another AXI4 Interconnect and configure it for one slave and three master ports.
2. Add an AXI Timer, AXI INTC (Interrupt Controller), and AXI UARTLite.

   Note: You can maximize the block diagram to increase the working area. This can be helpful in speeding things up. You can also float the block diagram window and scale it to a larger size. These buttons are located on the top left hand corner of the block diagram window.

Figure 43: Float/Maximize Block Diagram Window Buttons

3. Make the following connections:
   a. Connect the AXI Data Peripheral Interface (M_AXI_DP) of the MicroBlaze processor to the slave port S00_AXI of the AXI4 Interconnect that you added (axi_interconnect_2).
   b. Connect the S_AXI interface of the AXI Timer to a master port of the AXI4 Interconnect(2) that you added.
   c. Connect the S_AXI interface of the AXI INTC to one of the two remaining master ports of the AXI4 Interconnect(2).
   d. Connect the S_AXI interface of the AXI Uartlite to the remaining master port of the AXI4 Interconnect(2).
   e. Connect the clock from ui_clk port of mig_7_series to:
      - S_AXI_ACLK port and Processor_clk port of AXI INTC
      - S_AXI_ACLK port of AXI Timer
      - S_AXI_ACLK port of AXI Uartlite
      - All AXI4 Interconnect ACLK ports, all slave Sxx_ACLK ports, and all master Mxx_ACLK ports
   f. Connect the Peripheral_aresetn[0:0] port of proc_sys_reset to:
      - S_AXI_ARESETN port of AXI INTC
      - S_AXI_ARESETN port of AXI Timer
      - S_AXI_ARESETN port of AXI Uartlite
      - All AXI4 Interconnect slave Sxx_ARESETN ports and all master Mxx_ARESETN ports.
   g. Connect the MB_Reset port of proc_sys_reset to the Processor_rst port of AXI INTC.
   h. Connect the Interconnect_aresetn[0:0] port of proc_sys_reset to the ARESETN port of AXI4 Interconnect.
4. The connections you just made are shown in Table 4. Each row in the table represents a connection/net to be made). Refer to Figure 44 for these connections.

Table 4: **Peripheral Connections**

	AXI4 Interconnect_2	Mig 7 Series	AXI4 Timer	AXI4 INTC	AXI UARTLite	Proc_sys_ reset
1	M00_AXI		S_AXI			
2	M01_AXI				S_AXI	
3	M02_AXI			S_AXI		
4	ACLK S00_ACLK M00_ACLK M01_ACLK M02_ACLK	ui_clk	S_AXI_ACLK	Processor_clk S_AXI_ACLK	S_AXI_ACLK	
5	ARESETN					Interconnect _aresetn[0:0]
6	S00_ARESETN M00_ARESETN M01_ARESETN M02_ARESETN		S_AXI_ ARESETN	S_AXI_ ARESETN	S_AXI_ ARESETN	Peripheral_ aresetn[0:0]
7				Processor_rst		MB_Reset

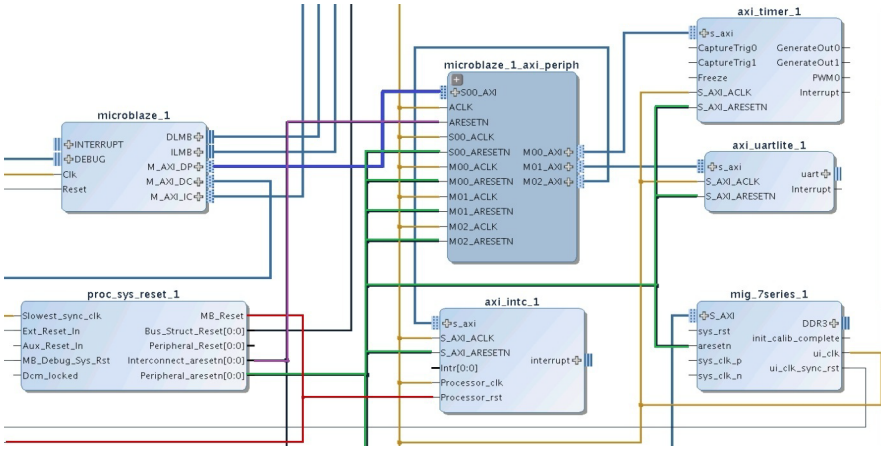

Figure 44: AXI4 Interconnect to IP Peripherals Connections

Add IP Concat

1. Add the concatenate tool, simply called Concat. This is used to "concatenate" the interrupt signals generated from AXI Timer and AXI UARTLite.
2. Connect the `Interrupt` port of AXI Timer to the input port `In0[0:0]` of Concat.
3. Connect the `Interrupt` port of AXI UARTLite to the input port `In1[0:0]` of Concat.
4. Connect the output pin `dout[1:0]` of Concat to the `Intr[0:0]` pin of AXI INTC.
5. Connect the `Interrupt` interface of the AXI INTC to the `INTERRUPT` interface of MicroBlaze processor.

   Figure 45 displays the connections for adding IP Concat.

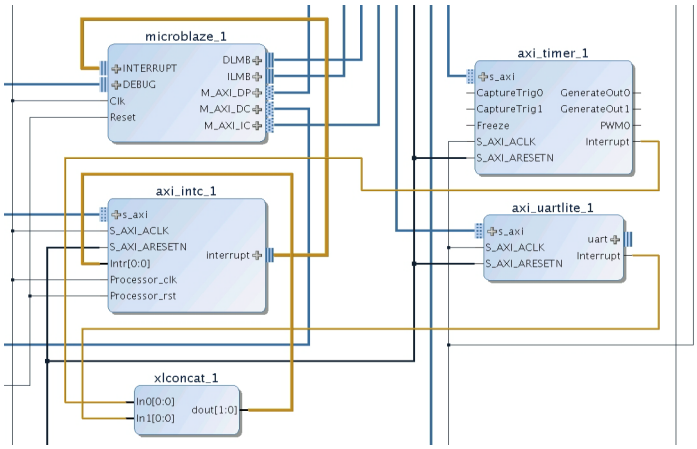

Figure 45: Adding IP Concat to Connect Interrupt Signals in the Design.

Configure the UARTLite interface

1. Click on the **+** sign next to the uart interface to expand it.

   The interface expands, showing two pins: `TX` and `RX`.

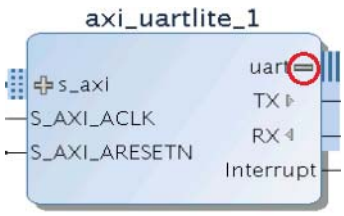

Figure 46: Expanding UART Interface Port of AXI UARTLite

2. Select the `TX` pin by clicking on it.

   Caution! Make sure that the port is selected when you create a new port. This ensures that the new port connects to the selected port. Alternately, you can create a port independently and manually create the connection to the pin.

3. Right-click on the `TX` pin while it is selected and select **Create Port**.

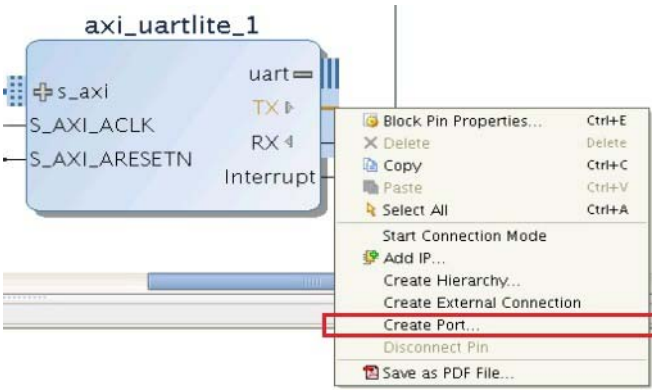

Figure 47: Creating a Port

4. In the create port options, select **Data** under Type.

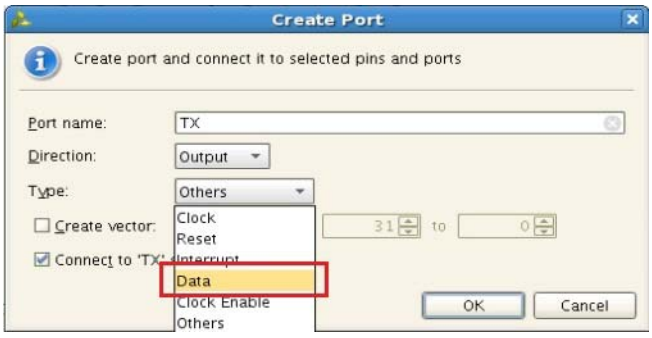

Figure 48: Configuring a Port During Creation

5. Click **OK**.

   A new port is added, as displayed in Figure 49.

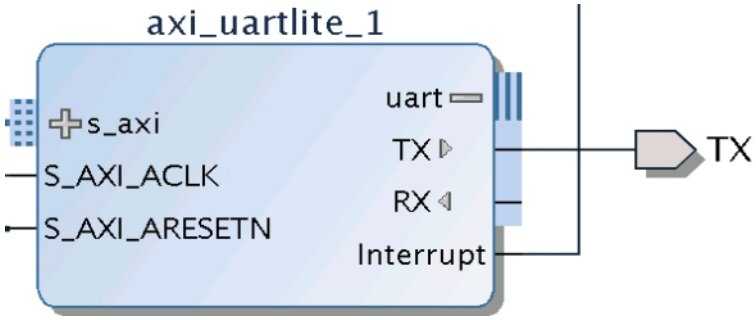

Figure 49: Port Created for TX

6. Create a port for `RX`. In the create port options, select the direction input and type as **Data**.

   Your AXI UARTLite IP will be similar to Figure 50.

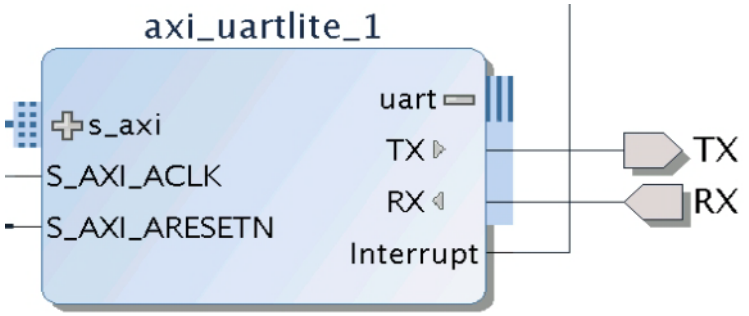

Figure 50: Created Port for RX

7. Alternately, you can do the following. To skip these steps, go to Creating Input Clock and Reset Ports for mig_7_series, page 30.

   a. Select the `uart` interface pin by clicking on it.

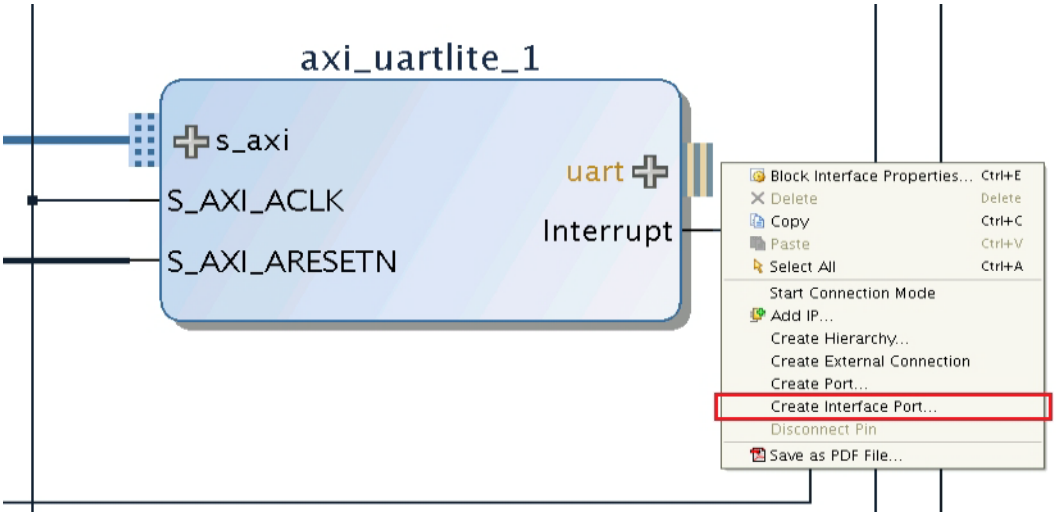

Figure 51: Creating an Interface Port

   b. Right-click on the pin and select **Create Interface Port**. This creates an interface port that will map all the ports in the interface.

Figure 52: Interface Port Settings

c. Click **OK** in the dialog box that opens.

An interface port is created. Select the interface port by clicking on it and you can see it listed as an External Interface in the design hierarchy tab and you can see the individual ports by expanding the interface by clicking on the **+** sign next to it.

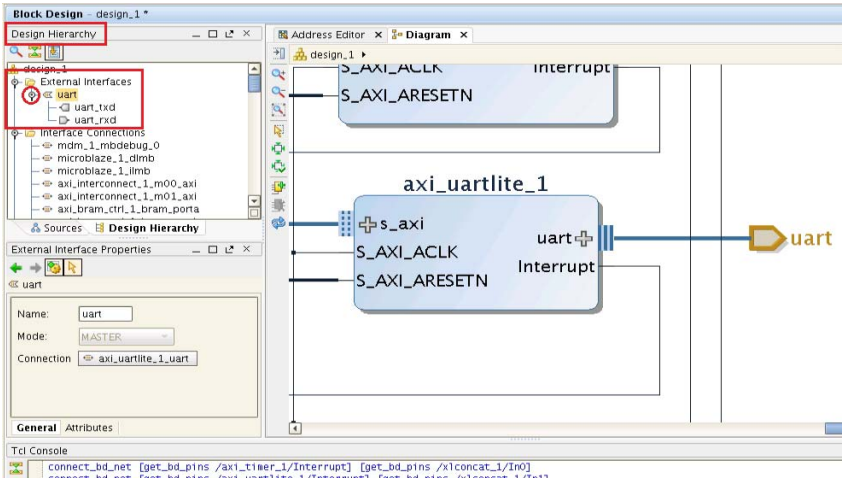

Figure 53: Viewing Individual Ports of the Interface Port

Creating Input Clock and Reset Ports for mig_7_series

1. Select the `sys_clk_p` port of mig_7_series by clicking on it.
2. Right-click the port and select **Create Port**.

Make sure that the port is selected when you create a new port. This ensures that the new port connects to the selected port. Alternately, you can create a port independently and manually create the connection to the pin.

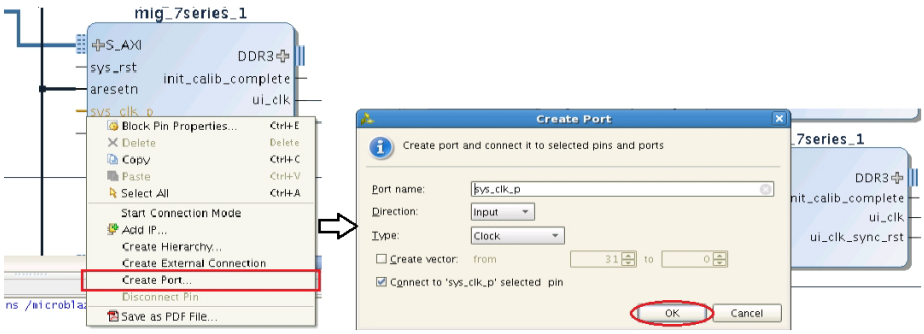

Figure 54: Creating Port for sys_clk_p Port of DDR

3. Create ports for `sys_clk_n` and `sys_rst` ports of mig_7_series.
4. Right-click the `DDR3` interface and select **Create Interface Port** to create an interface port. DDR now has the ports shown in Figure 55.

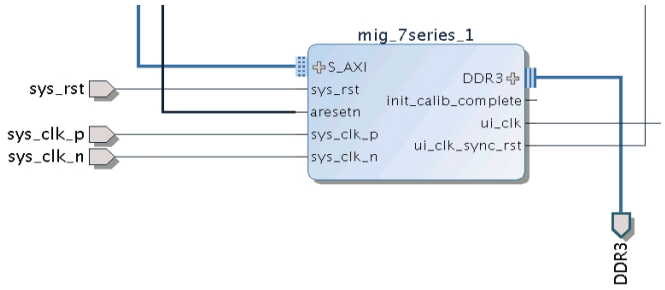

Figure 55: Mig_7_Series Ports

Creating Constraints

Your IP integrator block design is now complete.

Next, create a constraint (.xdc) file to map the ports that you created in the block diagram.

1. In the Flow Navigator tab of the Vivado IDE, go to the Project Manager.

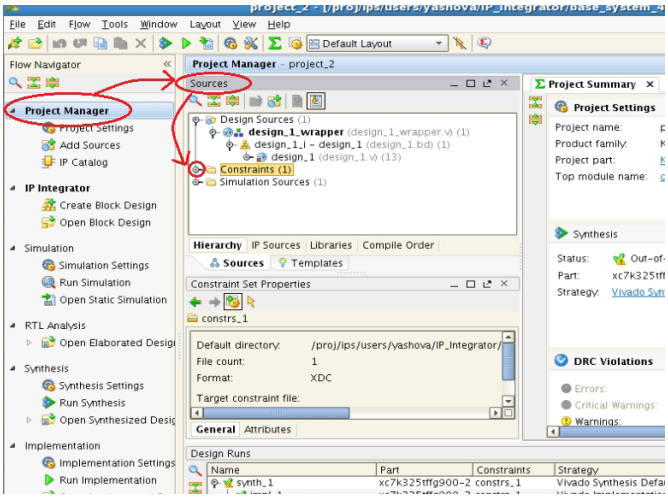

Figure 56: Add Constraints

2. Open the **Constraints** folder.

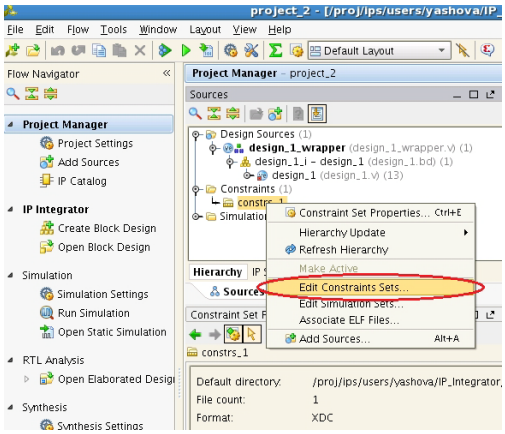

Figure 57: Edit Constraints Set

3. Right-click the constraints set and select **Edit Constraints Sets**.
4. In the Edit Constraints Set dialog box, click **Create File**.
5. Give your file a name and click **OK**.

   The Edit Constraints Set dialog box shows the new file.
6. Click **OK**.
7. Double click the file under the constraints set to open it.

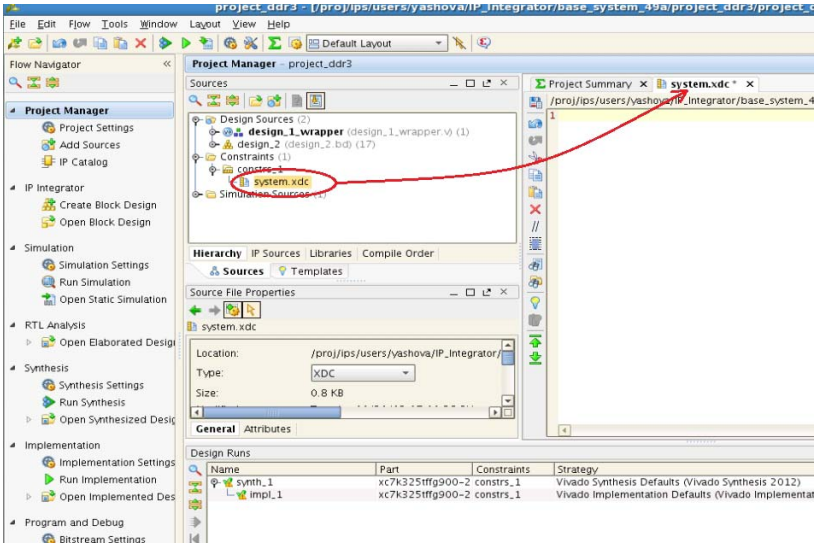

Figure 58: Open Constraints File

This file specifies local constraints for the ports created in the block diagram.

8. For the Kintex-7 FPGA, add the following code:

```
set_property LOC AD11 [ get_ports sys_clk_n]
set_property IOSTANDARD DIFF_SSTL15 [ get_ports sys_clk_n]
set_property LOC AD12 [ get_ports sys_clk_p]
set_property IOSTANDARD DIFF_SSTL15 [ get_ports sys_clk_p]
set_property LOC AB7 [ get_ports sys_rst]
set_property IOSTANDARD LVCMOS15 [ get_ports sys_rst]
set_property LOC M19 [ get_ports RX]
set_property IOSTANDARD LVCMOS25 [ get_ports RX]
set_property LOC K24 [ get_ports TX]
set_property IOSTANDARD LVCMOS25 [ get_ports TX]
#
# additional constraints
#
create_clock -name sys_clk_pin -period "5.0" [get_ports "sys_clk_p"]
# Added for RevC board
set_property slave_banks {32 34} [get_iobanks 33]
```

Caution! If you have created an interface port for UARTlite, replace the following in the above code:

`get_ports RX` to `get_ports uart_rxd`

`get_ports TX` to `get_ports uart_txd`

9. Save the file.

Mapping Addresses of Peripherals in an IP Integrator Design

1. In the Project Manager, double-click the design name under Design Sources to open the design.
2. Click the **Address Editor** tab.

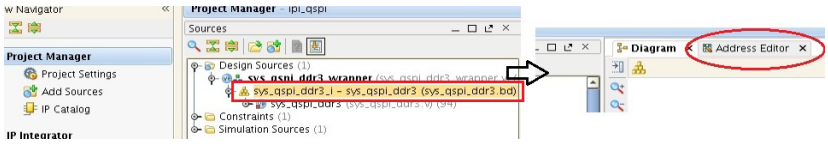

Figure 59: Address Editor

3. In the Address Editor, map any unmapped devices by doing the following:
   a. Expand the MicroBlaze IP.
   b. Right-click the **Unmapped Slaves** folder and select **Auto Assign Address**.

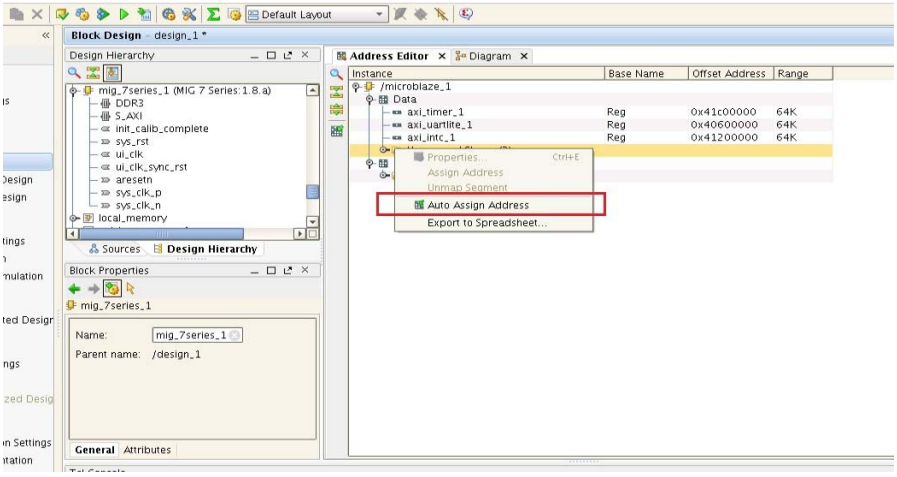

Figure 60: Auto Assign Memory Address to IPs

You can also change the address and range of the IPs.

4. Change the Range of mig_7_series IP in both the Data and the Instruction section to **512 MB**.
5. Save your design.

You must also ensure that the memory you are going to run and store your software from/in lies in the cacheable address range specified by assigning values to the cache(s) base address and cache(s) high address when you enabled Instruction Cache and Data Cache in re-configuring MicroBlaze processor.

In order to use either mig DDR or AXI block RAM, they must be in the cacheable area or MicroBlaze processor will not be able to read/write from them. Therefore your software won't work if you place it in these memories.

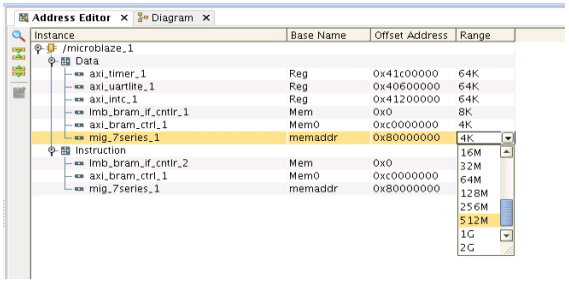

Figure 61: Assign Range to mig_7_series

You can also use this map to manually include or exclude IPs from the cacheable region or otherwise specify their address.

Creating a Top-Level Verilog Wrapper

1. Go to the Project Manager.
2. Under Design Sources, right-click your design and click **Create Top HDL**.

   A message notifies you that the block design file `<design_name>.bd` must be generated.
3. Click **Yes**.

   The dialog box that opens asks you to generate output products.
4. Click **OK**.

   Vivado creates a wrapper Verilog code for your design.

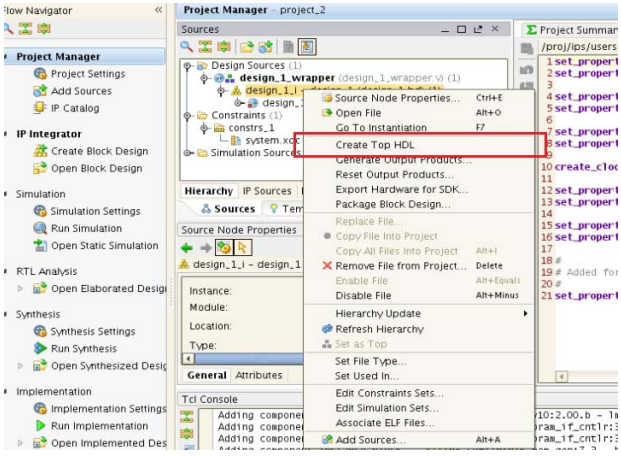

Figure 62: Create Top HDL

Reversing the Polarity of the Signal Coming to DDR Reset Input

Next, you must change the polarity of the `sys_rst` input that goes into the `sys_rst` port of DDR. To do this, you must make a few modifications in the `design_1_wrapper.v` file generated by the create top HDL function.

1. Double-click the top-level Verilog wrapper to open it.

   This is created over the block diagram in the Design Sources tab, as shown in Figure 63.

2. Change the line

   ```
   wire sys_rst ;
   ```

   to

   ```
   wire sys_rst_n;
   ```

3. After the line

   ```
   assign DDR_odt[0] = \^DDR_odt
   ```

   add the line

   ```
   assign sys_rst_n = ~sys_rst ;
   ```

4. Change the line

   ```
   .sys_rst(sys_rst);
   ```

   to

   ```
   .sys_rst(sys_rst_n);
   ```

5. Save the file and open the elaborated design

   Note: This step is optional; it is to check if the code entered has inverted the signal coming to sys_rst port. The Open Elaborated Design option is under the RTL Analysis header in the Flow Navigator tab.

   The system under the default layout has the input coming to the `sys_rst` port inverted, as shown in Figure 63.

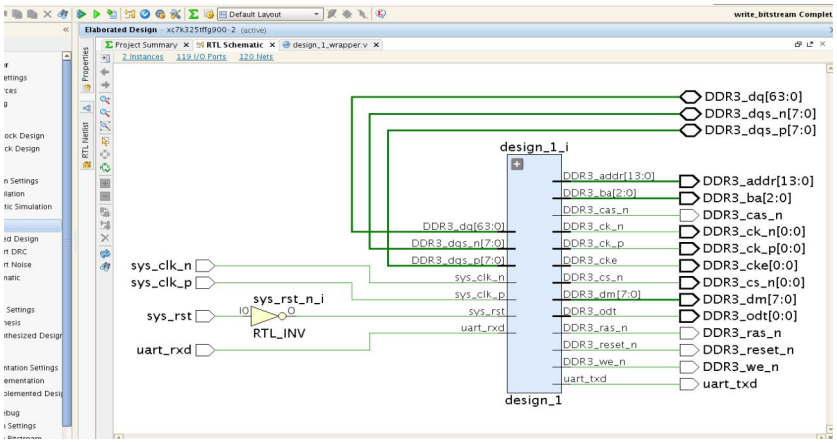

Figure 63: Elaborated Design - Default Layout

6. Run Synthesis.
7. Run Implementation.
8. Generate the Bitstream.

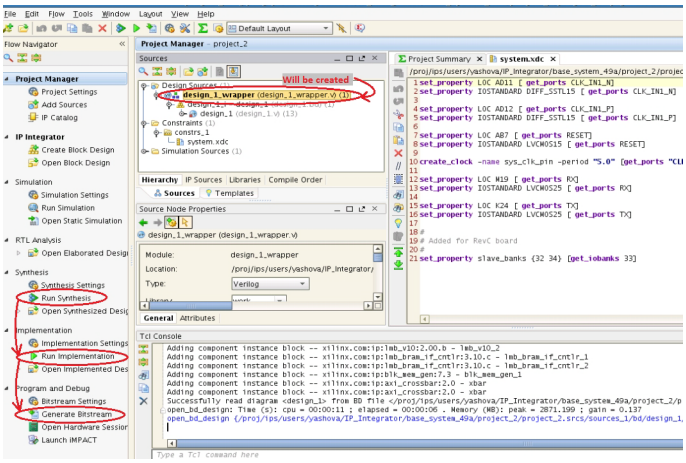

Figure 64: Run Synthesis, Run Implementation, and Generate the Bitstream

Software Reference System

Exporting the Design to SDK

Next, open the design and export to SDK.

1. Select **File > Export > Export Hardware for SDK**.
2. In the dialog box that opens, click to select the **Launch SDK** check box.

Figure 65: Select Workspace and Launch SDK

3. Specify the workspace to which you want to export your hardware design. If you don't specify the workspace, your project will be exported to `<project_name>.sdk/SDK/SDK_Export/` by default.
4. Click **OK**.

   SDK launches.

Configuring FreeRTOS

1. In SDK, select **Xilinx Tools > Repositories**.

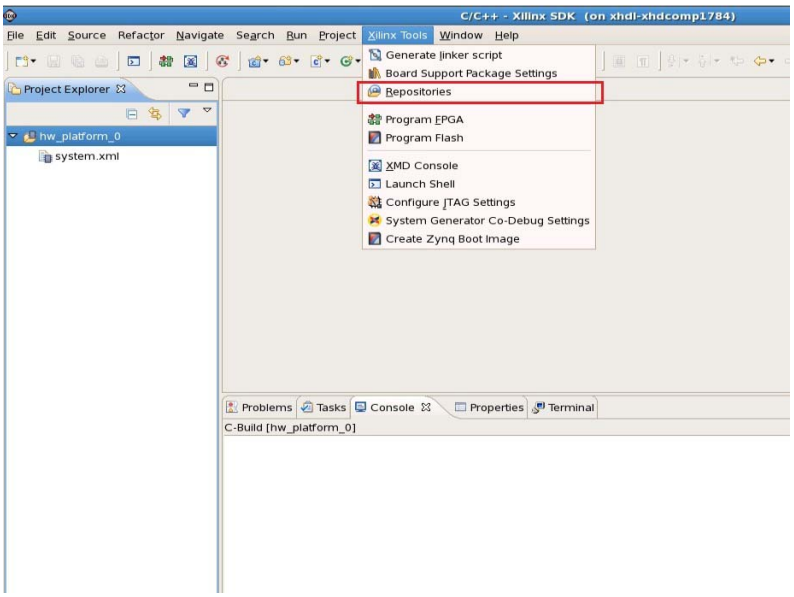

Figure 66: Configure Repositories to be Used by Project

2. Click on **New** next to the local repository box to add a new local repository.

   The freeRTOS repository is in the path `<reference design path>/IPI_DDR_XAPP/ipi_sys_ddr.sdk/sysrtos/lib`.
3. Add the above path and re-scan the repositories.
4. Apply settings and click **OK**.

Creating a freeRTOS "Hello World" Application.

1. In SDK, right-click **hw_platform_0** and select **New > Project**.
2. In the dialog box that opens, select **Xilinx Tools > Application Project**.
3. Click **Next**.
4. Type a name for your project and choose **FreeRTOS** as the OS platform.
5. Click **Next**.

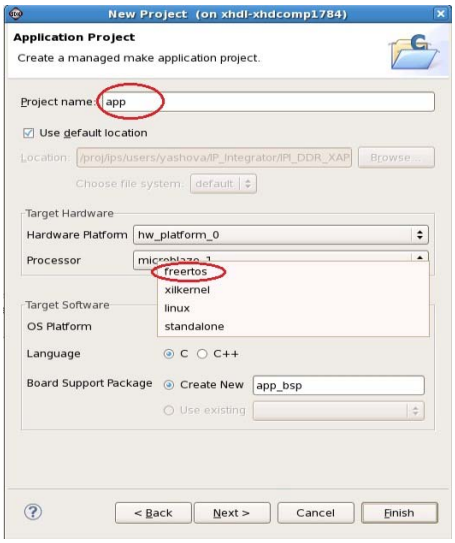

Figure 67: Select freeRTOS OS Platform and Name Your Project

6. Select the freeRTOS **Hello World** application template.
7. Click **Finish**.

   SDK creates a new Hello World application.

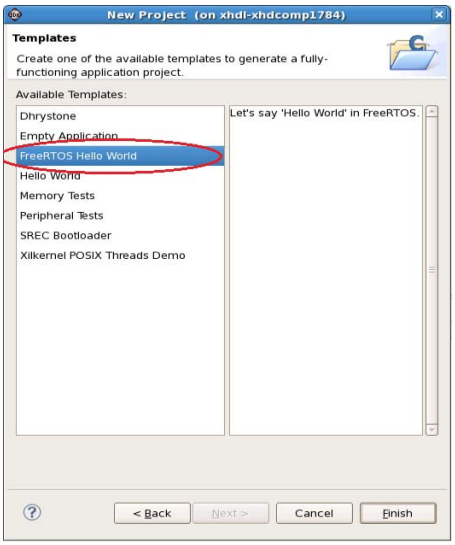

Figure 68: freeRTOS Hello World Application Template

A working software project can be found at `<reference design path>/IPI_DDR_XAPP/ipi_sys_ddr.sdk/sysrtos/`.

Executing the System on a KC705 Board

To run the design on a KC705 board:

1. Connect the board to the local device.
2. Open Tera Term.
3. In the menu bar, go to **Setup > Serial Port**, select the **COM** port for your system, and set the baud rate to **9600**.
4. Unzip the reference design to a desired location and go to the project directory.

   ```
   % cd <reference design path>/IPI_DDR_XAPP/
   ```

5. Start the xmd command line interpreter using the following command:

   ```
   % xmd
   ```

6. Burn the Bitstream on the board using the `fpga` command:

   ```
   % fpga -f ipi_sys_ddr/impl_1/sys_mb_ddr3_wrapper.bit
   ```

7. Start the MicroBlaze processor using one of the following commands:

   ```
   % connect mb mdm
   % mbc
   ```

8. Reset and stop the MicroBlaze processor before actually running the software by using the `rst` and `stop` commands, respectively.
9. Download the .elf file of the freeRTOS "Hello World" program.

   ```
   % dow ipi_sys_ddr.sdk/sysrtos/app/Debug/app.elf
   ```

   Note: The .elf is created by SDK and can be found in `<reference design path>/IPI_DDR_XAPP/ipi_sys_ddr.sdk/<path to your workspace>/<name of your application project>/Debug/<name of your application project>.elf`.

   The working .elf file provided with this project can be found in `<reference design path>/ipi_sys_ddr.sdk/sysrtos/app/Debug/app.elf`.

10. Run the program using the `run` command.

    ```
    % run
    ```

    Note: The .bit and .elf files can also be found at the location `<reference design path>/IPI_DDR_XAPP/ready_for_download`.

    The output will be displayed in Tera Term. The expected output is shown below.

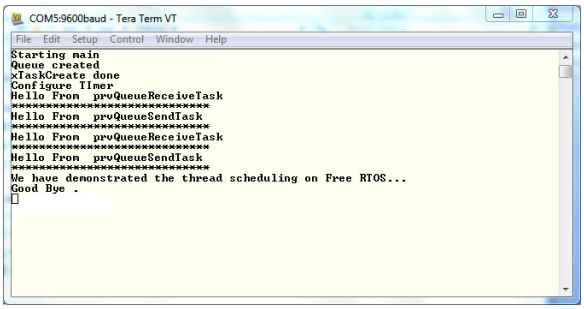

Figure 69: Expected Output

References

The current versions of the following documents are referenced in this document:

1. DS768, *LogiCORE IP AXI Interconnect Data Sheet*

Additional Information

The design files for this document are located at https://secure.xilinx.com/webreg/clickthrough.do?cid=201113.

The following documents provide additional information:

- UG883, *Kintex-7 FPGA KC705 Evaluation Kit Getting Started Guide*
- UG081, *MicroBlaze Processor Reference Guide*
- DS764, *LogiCORE IP AXI Timer Data Sheet*
- DS747, *LogiCORE IP AXI INTC Data Sheet*
- DS741, *LogiCORE IP AXI UART Lite Data Sheet*
- AMBA AXI4 specifications: http://infocenter.arm.com/help/topic/com.arm.doc.set.amba/index.html#specs
- FreeRTOS Porting Guide: http://www.freertos.org/FreeRTOS-porting-guide.html