

PLB v3.4 and OPB to PLB v4.6 System and Core Migration User Guide

UG443 (v1.2) December 14, 2007





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.

© 2007 Xilinx, Inc. All rights reserved. XILINX, the Xilinx logo, and other designated brands included herein are trademarks of Xilinx, Inc. PowerPC is a trademark of IBM, Inc. All other trademarks are the property of their respective owners.

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
8/29/07	1.0	Initial Xilinx release.
9/11/07	1.1	Added PPC405-PPC440 System Migration chapter.
12/14/07	1.2	Modified for EDK 10.1i.

Table of Contents

Preface: About This Guide

Guide Contents	5
Additional Resources	5
Conventions	5
Typographical	5
Online Document	6

Chapter 1: PLB v4.6 Overview

Introduction	7
Overview of PLB v4.6	7
Motivation and Advantages	7
PLB v4.6 Basics	7
New Features	9
Feature Restrictions	10
Signal and Specification Differences	11
Overview of PLBV46 IPIF Modules	13
PLBV46 Slave IPIF Modules	13
PLBV46 Master IPIF Modules	17
Master/Slave Mixed Data Width Considerations	22

Chapter 2: PPC405 - PPC440 System Migration

Introduction	27
System Migration Methodology	27
System Hardware Migration	27
System Software Migration	28
Existing PPC405 OPB/PLB v3.4 System	28
PowerPC 440/XPS System	29
System Migration Inside an EDK System	35
Migrating Buses/Processor	36
Migrating The Memory Controller	41
Migrating Ethernet Solution	42
Migrating Interrupt Controller to DCR INTC	45
Migrating XPS Peripherals	45
Modifying Clocking/Reset Inside The System	45
Migrating software from Virtex-4 PowerPC-405 to Virtex-5 PowerPC-440	64
Instruction Set Changes	64
Boot Vector	65
Interrupts	65
Caches	65
Memory Management	65
OCM	66
DCR	66
Timers	66
APU	67
Debugging	67

Chapter 3: PPC405 - PPC405/PLB v4.6 System Migration

Introduction	69
System Migration Methodology	69
System Hardware Migration	69
System Software Migration	70
Existing PPC405 PLB v3.4 System	70
PPC405 PLB v4.6/XPS System	71
Overview of System Migration	71
System Migration Inside an EDK System	73
Migrating Buses/Processor	73
Migrating The Memory Controller	76
Migrating Ethernet Solution	81
Migrating XPS Peripherals	84
Modifying Clocking/Reset Inside The System	85

Chapter 4: MicroBlaze System Migration

Introduction	97
System Migration Methodology	97
System Hardware Migration	97
System Software Migration	97
Example MicroBlaze OPB System	98
Migrated MicroBlaze PLB v4.6/XPS System	98
Overview of System Migration	99
System Migration Inside an EDK System	101
Migrating Buses/Processor	101
Migrating The Memory Controller	105
Migrating Ethernet Solution	110
Migrating XPS Peripherals	112
Modifying Clocking/Reset/Debug Inside The System	112

Chapter 5: Migration of User IP Slave Cores

Introduction	127
Overview of Create IP Wizard for OPB/PLBV34 Slaves	127
Create IP Wizard Slave Services	127
OPB and PLBV34 IPIFs	127
Overview of Create IP Wizard using PLB v4.6	128
Create IP Wizard Slave Services	128
PLBV46 Slave IPIFs	128
Overview of OPB/PLBV34 Slave Cores Created by Create IP Wizard	128
Creating PLBV46 Slave Cores with Create IP Wizard	128
Generating The PLBV46 Slave Cores Using Create IP Wizard	128
Parameters for PLBV46 Slave Cores	145
Modifying the Existing User Logic	146
Modifying Existing OPB User Logic	146
Modifying Existing PLBV34 User Logic	147

Chapter 6: Migration of User IP Master/Slave Cores

Introduction	149
---------------------------	-----

Overview of Create IP Wizard for OPB/PLBV34 Master/Slaves	149
Create IP Wizard Master Services	149
OPB/PLBV34 IPIFs	149
Overview of Create IP Wizard for PLB v4.6 Master/Slaves	150
Create IP Wizard Master Services	150
PLBV46 Master/Slave IPIFs	150
Overview of OPB/PLBV34 Master/Slave Cores Created By Create IP Wizard	150
Creating the PLBV46 Master/Slave Cores Using Create IP Wizard	151
Generating the PLBV46 Master/Slave Cores	151
Parameters for PLBV46 Master/Slave Cores	165

About This Guide

Guide Contents

This manual contains the following chapters:

[Chapter 1, "PLB v4.6 Overview"](#)

[Chapter 2, "PPC405 - PPC440 System Migration,"](#)

[Chapter 3, "PPC405 - PPC405/PLB v4.6 System Migration"](#)

[Chapter 4, "MicroBlaze System Migration"](#)

[Chapter 5, "Migration of User IP Slave Cores"](#)

[Chapter 6, "Migration of User IP Master/Slave Cores"](#)

Additional Resources

To find additional documentation, see the Xilinx website at:

<http://www.xilinx.com/literature>.

To search the Answer Database of silicon, software, and IP questions and answers, or to create a technical support WebCase, see the Xilinx website at:

<http://www.xilinx.com/support>.

Conventions

This document uses the following conventions. An example illustrates each convention.

Typographical

The following typographical conventions are used in this document:

Convention	Meaning or Use	Example
Courier font	Messages, prompts, and program files that the system displays	<code>speed grade: - 100</code>
Courier bold	Literal commands that you enter in a syntactical statement	ngdbuild <i>design_name</i>

Convention	Meaning or Use	Example
Helvetica bold	Commands that you select from a menu	File → Open
	Keyboard shortcuts	Ctrl+C
Italic font	Variables in a syntax statement for which you must supply values	<code>ngdbuild design_name</code>
	References to other manuals	See the <i>Development System Reference Guide</i> for more information.
	Emphasis in text	If a wire is drawn so that it overlaps the pin of a symbol, the two nets are <i>not</i> connected.
Square brackets []	An optional entry or parameter. However, in bus specifications, such as <code>bus [7:0]</code> , they are required.	<code>ngdbuild [option_name] design_name</code>
Braces { }	A list of items from which you must choose one or more	<code>lowpwr = {on off}</code>
Vertical bar	Separates items in a list of choices	<code>lowpwr = {on off}</code>
Vertical ellipsis . . .	Repetitive material that has been omitted	IOB #1: Name = QOUT' IOB #2: Name = CLKIN' . . .
Horizontal ellipsis ...	Repetitive material that has been omitted	<code>allow block block_name loc1 loc2 ... locn;</code>

Online Document

The following conventions are used in this document:

Convention	Meaning or Use	Example
Blue text	Cross-reference link to a location in the current document	See the section “ Additional Resources ” for details. Refer to “ Title Formats ” in Chapter 1 for details.
Red text	Cross-reference link to a location in another document	See Figure 2-5 in the <i>Virtex-II Platform FPGA User Guide</i> .
Blue, underlined text	Hyperlink to a website (URL)	Go to http://www.xilinx.com for the latest speed files.

PLB v4.6 Overview

Introduction

This chapter provides an overview of the Xilinx PLB v4.6 interconnect as it applies to the migration of User IP cores and systems. The full specification of the Xilinx PLB v4.6 interconnect is not provided in this document, but the differences from existing OPB/PLB v3.4 interconnects are highlighted and an overview of the PLBV46 IP Interface (IPIF)s is provided. It is assumed that the reader has working knowledge of the existing Xilinx implementations of OPB/PLB v3.4 interconnects.

Overview of PLB v4.6

This section will provide an overview of the Xilinx PLB v4.6 interconnect, but will not provide the detailed specification. Please refer to SP026 PLBV46 Interface Simplifications for more details.

Motivation and Advantages

The prime motivation to develop another interconnect for embedded systems and migrate all Processor IP Cores and systems to this interconnect scheme is to increase system performance. Years of experience developing the existing Xilinx embedded solution provided insight into the features and modifications necessary to provide an interconnect solution with improved performance while still preserving the development infrastructure.

Another motivation was improving the ease of use when developing Embedded Systems. Migrating the Xilinx Processor IP Catalog to PLB v4.6/XPS reduces the number of different cores available that provide the same basic function and allows the same IP and system architecture to be used across processors. This greatly simplifies the system design and implementation task because the user doesn't have to determine which cores can be in which configuration and with the specific processor of choice.

PLB v4.6 Basics

Address and Data Buses

PLB v4.6 supports a 32-bit address bus. The read and write data busses are split and can either be 32-bit, 64-bit, or 128-bit data width. Though the arbitration logic supports a 2-deep address pipeline, this is currently not implemented in the Xilinx PLB v4.6/XPS soft IP slaves.

Address and Data Phases

PLB v4.6 transactions are split into an address phase and a data phase. The address phase is when the requested address and transaction qualifiers are driven to all slaves. It starts with the assertion of PLB_PAVald and terminates with the assertion of Sl_AddrAck. Address Phases may overlap (in time).

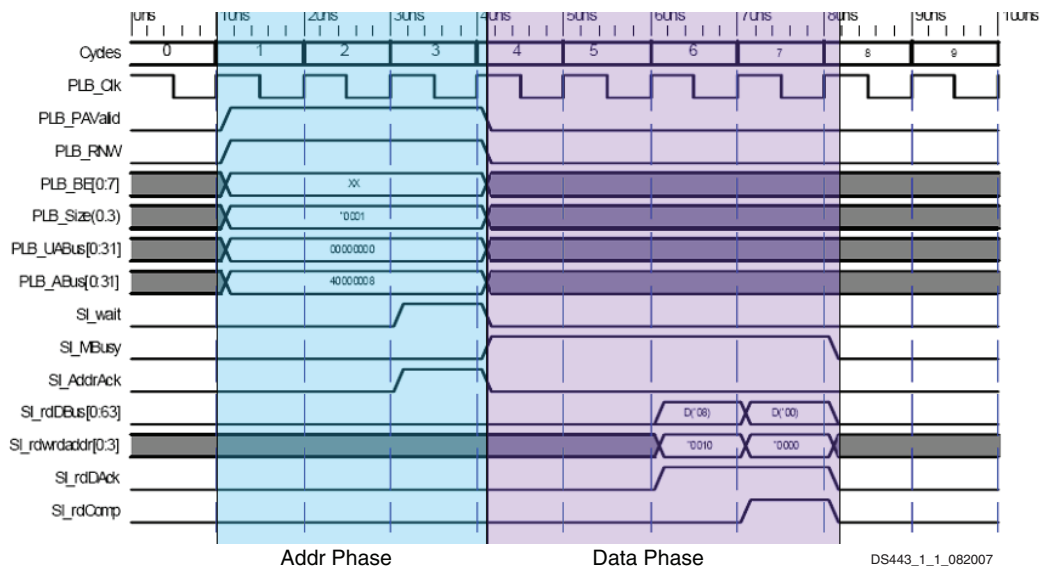


Figure 1-1: PLBV46 Address Phase and Data Phase

The data phase is when the actual data transfer occurs. A read data phase and a write data phase can occur simultaneously. The data phase starts with the assertion of Sl_AddrAck and ends with the assertion of Sl_wrComp or Sl_rdBComp. A read data phase and write data phase may overlap.

Transactions

PLB v4.6 supports single data beat, cacheline, and fixed length burst of 2 to 16 data beats. Word (32-bit), double-word (64-bit), and quad-word (128-bit) data widths are allowed for burst transactions. Fixed Length Burst requests with data beat counts from 17 to 256 and transfer widths of bytes or half words will be ignored by Xilinx soft IP Slaves. Cacheline transactions are 4-word, 8-word, or 16-word transfers. Target-word first support requires circular address generation in the Slave.

PLB_Size communicates information about transaction data width, type, and length as shown below:

- ◆ "0000" = Single data beat, PLB_BE indicates number of bytes
- ◆ "0001" = 4-word Cacheline
- ◆ "0010" = 8-word Cacheline
- ◆ "0011" = 16-word Cacheline
- ◆ "0100" – "0111", "1110" – "1111" = Reserved
- ◆ "1010" = Word Burst Transfer
- ◆ "1011" = Double Word Burst Transfer

- ◆ “1100” = Quad Word Burst Transfer
- ◆ “1101” = Octal Word Burst Transfer

When PLB_Size is indicating a Burst Transfer, the length of the burst is specified by PLB_BE. Note that a zero value on PLB_BE when specifying the burst length specifies an indeterminate length burst which is not supported in the Xilinx PLB v4.6 interconnect.

Timeouts

The PLB v4.6 implements an address phase timeout of 16 clocks. The timeout counter (located in the arbiter) can be suspended by assertion of SI_Wait by the targeted slave. PLB v4.6 data phases have no timeout limit, however, the PLBV46 IPIFs implement a timeout counter from 0 to 511 clocks.

Aborts

Transactions can not be aborted in the Xilinx PLB v4.6 interconnect.

New Features

The Xilinx PLB v4.6 interconnect optimizes the existing PLB v3.4 interconnect by removing unneeded features to reduce resource utilization and lead to higher Fmax.

Point-to-Point Bus Topology

Xilinx PLB v4.6 interconnect supports a Point-to-Point bus topology for configurations that contain a single master and single slave. This feature eliminates the need for the slave to perform address decode, therefore, the resources required by the slave are reduced. Also, the latency of the transaction is reduced since the address phase simplifies to a single clock due to the elimination of arbitration and address decode. Latency is further reduced due to the elimination of interface registers in the master and slave. Since the master and slave outputs only connect to a single load, the need for interface registers is removed.

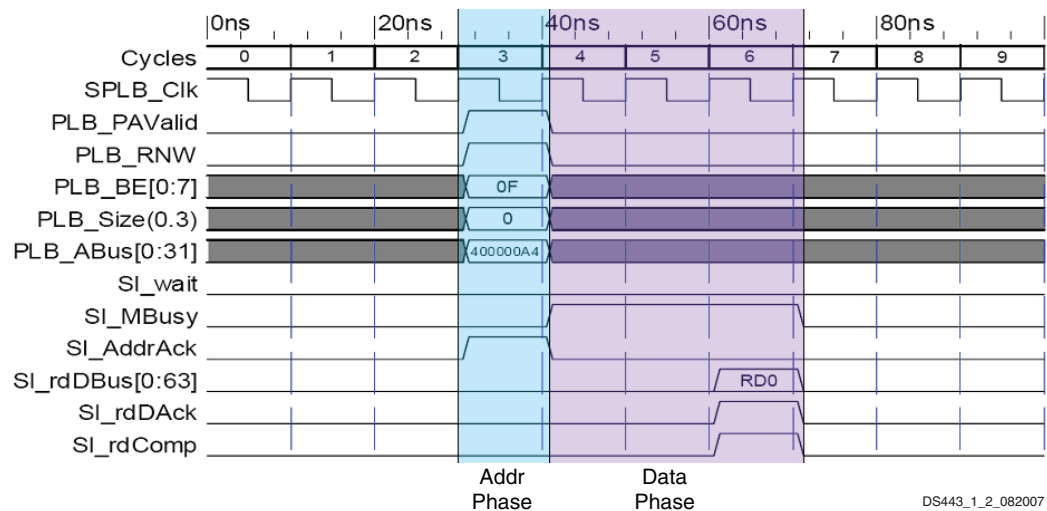


Figure 1-2: Reduced Address Phase when using Point-2-Point Mode

Feature Subsets

Modes were added to streamline IP to use only the features needed by that IP. Two feature subsets of the IBM 128-bit Processor Local Bus Architectural Specification (v4.6) are implemented: the Baseline feature subset and the Performance feature subset.

Note: The PLB v4.6 Interface signal set is the same for peripherals which implement either of the feature subsets.

Baseline Feature Subset

The Baseline feature subset is mainly targeted at PLB v4.6 slaves that contain low-speed data and/or control registers. The peripherals that would use the Baseline feature subset are always 32-bit devices which never support burst transactions. Data steering is never required. In summary, the Baseline feature subset provides the following:

- ◆ Single unit transfers (no bursts or cache-line)
- ◆ 32-bit address (4 GB space)
- ◆ 32 bit wide data transfers
- ◆ Interface is required to connect to 32, 64 or 128-bit wide data buses (per IBM 128-bit Processor Local Bus Architectural Specification (v4.6))

Performance Feature Subset

The Performance feature subset was developed to support high-bandwidth masters and slaves. There is no restriction on the data width of the performance peripherals. They can be 32, 64, or 128 bits wide. However, data mirroring (masters) or data steering (slaves) logic may be required when communicating with narrower devices. Bursts transactions up to 16 beats are supported. Master conversion cycles are supported and allow burst transactions to narrower slaves. In summary, the Performance feature subset provides the following:

- ◆ Fixed-length bursts up to 16 data beats
- ◆ Cache-line transfers
- ◆ 32, 64 or 128 bit wide data transfers
- ◆ Connects to same or larger width buses
- ◆ Master data mirroring for writes to narrower slaves
- ◆ Slave data steering for reads to narrower masters

Feature Restrictions

To optimize the resource utilization and performance of Xilinx PLB v4.6 peripherals, unused and unneeded features of the full IBM 128-bit Processor Local Bus Architectural Specification (v4.6) have been eliminated and restrictions placed on the implementation of some features.

The following features of the IBM 128-bit Processor Local Bus Architectural Specification (v4.6) are not supported:

- ◆ Requests for bursts longer than 16 data-beats
- ◆ Indeterminate length burst transfers
- ◆ Aborts (suppressed in PPC processor)

The following feature restrictions are in place:

- ◆ Master burst requests to non-bursting (baseline) slaves are prohibited
- ◆ Fixed-length bursts cannot be prematurely terminated by a master or a slave
- ◆ Only memory transfer types (M_type[0:2]="000") are supported

Note: The above lists are not all-inclusive. Please refer to SP026 PLBV46 Interface Simplifications for complete details.

Signal and Specification Differences

The Xilinx PLB v4.6 interconnect signal set is basically the same as the existing Xilinx PLB v3.4 interconnect signal set. The basic protocols and transactions are also the same as the existing Xilinx PLB v3.4 basic protocols and transactions. Therefore, for users familiar with the existing Xilinx PLB v3.4 interconnect, there is little or no learning curve.

This section will highlight the major differences in the implemented signals and transactions from the IBM 128-bit Processor Local Bus Architectural Specification (v4.6).

Note: This section will not exhaustively list all differences and will not be all-inclusive. Please refer to SP026 PLBV46 Interface Simplifications for complete details.

Upper Address Bus and Parity Signals

The IBM 128-bit Processor Local Bus Architectural Specification (v4.6) specification added an upper address bus and optional parity and parity enable signals for the address busses, byte enable busses, and data busses but are not supported in Xilinx soft IP:

- ◆ Mn_UABus(0:31), PLB_UABus(0:31)
- ◆ Mn_ABusPar, Mn_ABusParEn, Mn_UABusPar, Mn_UABusParEn
- ◆ PLB_ABusPar, PLB_ABusParEn, PLB_UABusPar, PLB_UABusParEn
- ◆ Mn_BEPar, Mn_BEParEn, PLB_BEPar, PLB_BEParEn
- ◆ Mn_wrDBusPar, Mn_wrDBusParEn, PLB_wrDBusPar, PLB_wrDBusParEn
- ◆ PLB_MnRdDBusPar, PLB_MnRdDBusParEn, Sl_rdDBusPar, Sl_rdDBusParEn

In addition, parity error signals were also added:

- ◆ Sl_ABusParErr, Mn_rdDBusParErr

Note: These signals are not supported by Xilinx soft IP.

Interrupt Indicators

A set of signals to allow interrupts to be signalled between masters and slaves is specified in the IBM 128-bit Processor Local Bus Architectural Specification (v4.6) but are not supported in Xilinx soft IP:

- ◆ PLB_MIRQ(n), Sl_MIRQ(0:n)

Note: These signals are not supported by Xilinx soft IP.

Master Bus Timeout

A timeout indicator is now available for each master on the PLB v4.6 interconnect. This allows the master to distinguish between when a transaction has timed out and a transaction that completes with an error. The master timeout signal is sent from the arbitration logic:

- ◆ PLB_MnTimeout

Pending Request Signals

The Pending Request signals are now split into pending requests for read and write transactions but are ignored by Xilinx soft IP:

- ◆ PLB_pendReq => PLB_rdPendReq, PLB_wrPendReq
- ◆ PLB_pendPri(0:1) => PLB_rdPendPri(0:1), PLB_wrPendPri(0:1)

Note: These signals are ignored by Xilinx soft IP

Error Indicators

Error indicators are split into separate read and write signals in PLB v4.6 and are supported by Xilinx soft IP:

Sl_MErr(0:n) => Sl_MRdErr(0:n), Sl_MWrErr(0:n)

PLB_Mn_Err => PLB_MRdErr(n), PLB_MWrErr(n)

Compressed, Guarded, and Ordered Signals

The compressed, guarded, and ordered signals have been removed. This information is now conveyed via the Transfer Attribute bus, but are not supported by Xilinx soft IP:

- ◆ Mn_compress, Mn_guarded, Mn_ordered => Mn_TAttribute(0:15)
- ◆ PLB_compress, PLB_guarded, PLB_ordered => PLB_TAttribute

Note: These signals are not supported by Xilinx soft IP.

Xilinx Soft IP Support Summary

The Xilinx implementation of the PLB v4.6 interconnect in soft IP doesn't include support for all of the signals specified in the specification. Here's a summary of the support limitations:

- ◆ Optional parity signals not supported
- ◆ Only memory transfers supported
- ◆ M_type, PLB_type = "000"
- ◆ Transfer attributes not supported
- ◆ Pending request input status signals are present on the Master/Slave interfaces, but are ignored
- ◆ Slave to Master interrupts are not supported

Overview of PLBV46 IPIF Modules

This section will provide an overview of the PLBV46 IPIF modules. For more detailed information, the user is encouraged to reference the IP Data sheet for the PLBV46 IPIF of interest.

There are four PLBV46 IPIF modules available:

- ◆ PLBV46 Slave Single
- ◆ PLBV46 Slave Burst
- ◆ PLBV46 Master Single
- ◆ PLBV46 Master Burst

These four PLBV46 IPIFs provide a slave and master version of the PLB v4.6 baseline and the PLB v4.6 performance feature subsets. Note that the master and slave functions are now completely separated, unlike the existing OPB and PLBV34 IPIF modules. All of the PLBV46 master and slave IPIF modules provide support for point-to-point configurations.

The services provided by the existing OPB and PLBV34 IPIF modules are no longer included in any of the PLBV46 IPIF modules. The services are available as separate libraries and are still easily accessible for inclusion in User IP cores through the Create / Import IP Wizard.

PLBV46 Slave IPIF Modules

PLBV46 Slave Single IPIF Module

The PLBV46 Slave Single IPIF module implements the PLB v4.6 baseline feature subset for a slave peripheral. Its main target use is for register accesses within user IP. [Figure 1-3](#) shows the block diagram for the PLBV46 Slave Single IPIF module.

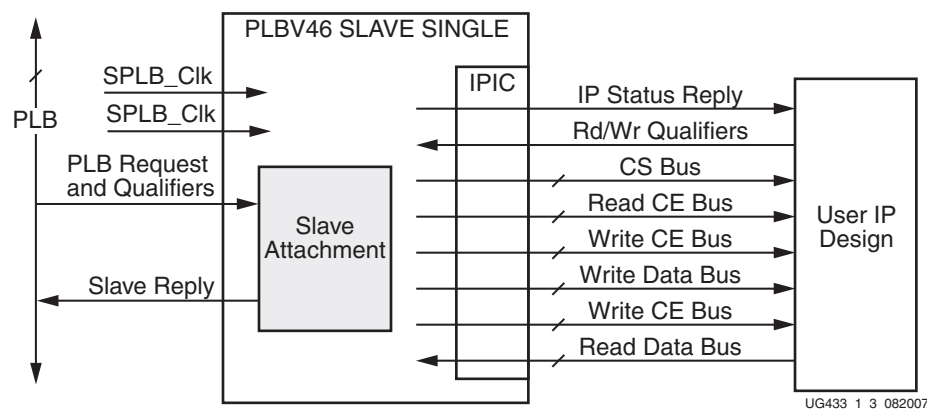


Figure 1-3: PLBV46 Slave Single Block Diagram

The PLBV46 Slave Single IPIF module supports a 32-bit address bus and a 32-bit data bus. Note that 64-bit and 128-bit data widths are not supported, therefore, a core utilizing this IPIF can only have a data-width of 32-bit. However, cores utilizing this IPIF can connect to PLBV46 busses of 64-bits and 128-bits as required by the Xilinx PLB v4.6 specification.

Only single read and write data transfers are supported by the PLBV46 Slave Single IPIF module. There is no support for burst transfers with this IPIF. If a burst transaction is

addressed to a slave utilizing this IPIF, the address acknowledge will never be asserted causing a bus timeout.

The PLBV46 Slave Single IPIF is the only PLBV46 IPIF that supports allowing the User IP to run from a separate, slower clock than the PLB v4.6 bus clock. (This feature is not available in the PLBV46 Slave Burst, PLBV46 Master Single, or PLBV46 Master Burst IPIF modules). The PLBV46 Slave Single IPIF supports a 1:1 and a 2:1 clock ratios which are controlled by the parameter C_BUS2CORE_CLK_RATIO as shown in Table 1-1.

Table 1-1: PLBV46 Slave Single C_BUS2CORE_CLK_RATIO

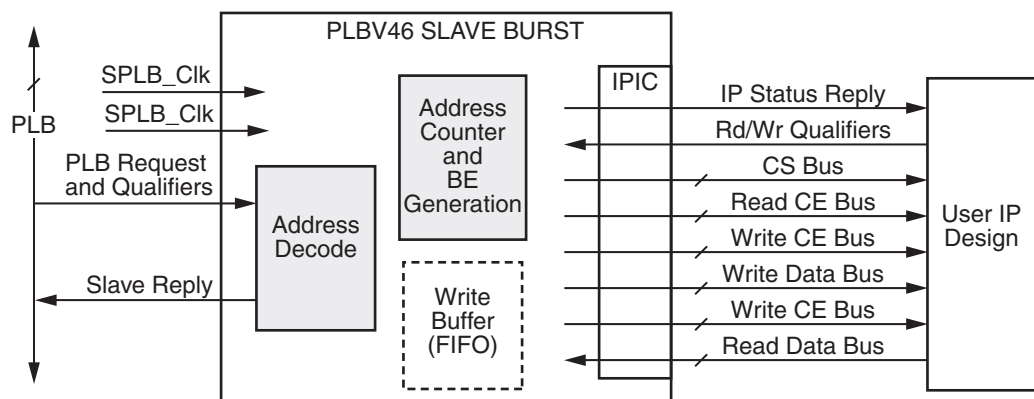
C_BUS2CORE_CLK_RATIO Value	Resulting Bus to Core Clock Ratio
1	1:1
2	2:1

Note: C_BUS2CORE_CLK_RATIO is not configurable through the Create IP Wizard.

When C_BUS2CORE_CLK_RATIO = 2, synchronization registers are included in the PLBV46 Slave Single IPIF module. Note, however, that the PLBV46 Slave Single IPIF module will not create the clock for the User Logic, this clock must be connected directly to the User Logic. It will not route through the IPIF and the User Logic should not connect to the Bus2IP_Clk output from the IPIF. Also note that the User Logic clock and the PLB v4.6 bus clock must be edge synchronous, i.e., must be outputs from the same DCM.

PLBV46 Slave Burst IPIF Module

The PLBV46 Slave Single IPIF module implements the PLB v4.6 performance feature subset for a slave peripheral. Its main target use is for higher-throughput slaves such as memory controllers and bridges. Figure 1-4 shows the block diagram for the PLBV46 Slave Burst IPIF module.



Note:
1. Items in broken line outline are optional service modules.

UG433_1_4_082007

Figure 1-4: PLBV46 Slave Burst Block Diagram

The PLBV46 Slave Burst IPIF module supports a 32-bit address bus and a 32-bit, 64-bit, or 128-bit data bus. Cores utilizing this IPIF can connect to PLB v4.6 busses of 32-bits, 64-bits and 128-bits as required by the Xilinx PLB v4.6 specification.

Single read and write data transfer, fixed length burst transfers (up to 16 data beats), and cacheline transfers are all supported by the PLBV46 Slave Burst IPIF module. The cacheline read transfer response can be configured to be either target-word first or line word first.

PLBV46 Slave IPIF Parameters

The PLBV46 Slave IPIF modules have parameters that are not available in the existing OPB and PLBV34 IPIF modules. These parameters are discussed in detail in the Migration of User IP Slave Cores chapter. The user is encouraged to refer to the data sheets for the PLBV46 Slave IPIF modules for more details.

Data width parameters should be given special consideration as to the system that this IP will be used in. When these parameters are of mis-matched values, additional logic within the slave IP may be required in order to support burst run length expansion, byte-enable muxing, and read data bus steering. Please refer to the section, “[Master/Slave Mixed Data Width Considerations](#)” of this chapter for more information.

PLBV46 Slave IPIC Signal Comparison

The following tables show comparisons of the IPIC signal sets for the existing OPB and PLBV34 IPIF modules and the PLBV46 Slave IPIF modules. Please refer to the data sheets for the PLBV46 Slave IPIF module of interest for more details and descriptions of these signals.

The signals from the IPIF modules to the User Logic (Bus2IP) are compared in [Figure 1-5](#).

OPB	PLBV34	PLBV46
Bus2IP_Data	Bus2IP_Data	Bus2IP_Data
Bus2IP_Addr	Bus2IP_Addr	Bus2IP_Addr
Bus2IP_RNW	Bus2IP_RNW	Bus2IP_RNW
Bus2IP_BE	Bus2IP_BE	Bus2IP_BE
Bus2IP_Burst	Bus2IP_Burst	Bus2IP_Burst (Only for PLBV46 Slave Burst)
Bus2IP_WrReq	Bus2IP_WrReq	Bus2IP_WrReq (Only for PLBV46 Slave Burst)
Bus2IP_RdReq	Bus2IP_RdReq	Bus2IP_RdReq (Only for PLBV46 Slave Burst)
Bus2IP_CS	Bus2IP_CS	Bus2IP_CS
Bus2IP_CE	Bus2IP_CE	
Bus2IP_RdCE	Bus2IP_RdCE	Bus2IP_RdCE
Bus2IP_WrCE	Bus2IP_WrCE	Bus2IP_WrCE
	Bus2IP_IBurst	
	Bus2IP_Abort	
		Bus2IP_BurstLength (Only for PLBV46 Slave Burst)

UG443_1_5_082007

Figure 1-5: Comparison of Slave IPIC Signals (Bus2IP)

Note that the Bus2IP_CE vector is no longer available in the PLBV46 Slave IPIFs. This vector was redundant with the Bus2IP_RdCE and Bus2IP_WrCE vectors. Because indeterminate bursts and aborts are no longer supported in the Xilinx PLB v4.6 interconnect, the signals Bus2IP_IBurst and Bus2IP_Abort have been removed.

Since only fixed length bursts are supported by PLBV46 Slave Burst IPIF modules, the signal, Bus2IP_BurstLength, has been added to supply the User Logic of the IP with the length of the current burst transaction. Knowledge of the transaction burst length may provide the opportunity to optimize the User Logic.

The signals from the User Logic to the IPIF module are compared in [Figure 1-6](#).

OPB	PLBV34	PLBV46
IP2Bus_Data	IP2Bus_Data	IP2Bus_Data
IP2Bus_WrAck	IP2Bus_WrAck	IP2Bus_WrAck
IP2Bus_RdAck	IP2Bus_RdAck	IP2Bus_RdAck
IP2Bus_Retry	IP2Bus_Retry	
IP2Bus_Error	IP2Bus_Error	IP2Bus_Error
IP2Bus_ToutSup	IP2Bus_ToutSup	
IP2Bus_PostedWrInh		
	IP2Bus_Busy	
	IP2Bus_AddrAck	IP2Bus_AddrAck (Only for PLBV46 Slave Burst)
	IP2Bus_Bterm	

UG443_1_6_082007

Figure 1-6: Comparison of Slave IPIC Signals (IP2Bus)

IP2Bus_Retry had limited use in existing IP. It originated in the OPB IPIF so that User IP could control the Sln_Retry signal on the OPB. Assertion of this signal terminated the OPB transaction without transferring data and informed the Master that the transaction could be tried again later. The PLBV34 and PLBV46 buses do not have a retry signal. It was present on the IPIC of the PLBV34 IPIFs simply for compatibility with the OPB IPIC. Most processor IP did not utilize this signal, therefore, it was removed from the PLBV46 IPIC signal set. If the User IP Logic requires a method of communicating to the master that it is busy with another operation, this is best implemented by means of a status register.

IP2Bus_ToutSup also originated in the OPB IPIF so that User IP could control the Sln_ToutSup signal on the OPB to suppress the 16-clock timeout counter in the OPB arbiter. This signal is not present on the PLBV34 or PLBV46 buses. It was present in the PLBV34 IPIC, however, to suppress the 64-clock timeout counter present in the PLBV34 IPIF to terminate a data phase transaction that wasn't acknowledged by the User IP. The PLBV46 IPIF now contains a 128-clock data phase timeout counter. With the much longer timeout value, it was deemed unnecessary to provide a suppression capability, therefore, this signal is not present in the IPIC for PLBV46.

IP2Bus_PostedWrInh added huge complexities for OPB IPIFs that didn't seem to merit the slight additional flexibility in user IP, therefore, this signal is no longer available.

IP2Bus_Busy was predominantly used in Master/Slave combinations to indicate to the slave attachment that the master attachment was utilizing the slave attachment and therefore it could not accept any bus transactions. For PLBV46 IPIF modules, the master and slave modules are now completely separate. If there are portions of the User IP logic that need to be protected from bus transactions during certain operations of the accompanying master logic, the User Logic must accommodate this in the User Logic

design as there is no longer a busy indicator from the User Logic to the PLBV46 Slave IPIF module.

IP2Bus_Bterm has been removed because the early termination of a burst is no longer allowed.

IPBus_AddrAck allows the User Logic to request addresses separately and ahead of acknowledging data. This is only useful during burst transactions and therefore this signal is only supported in PLBV46 Slave Burst IPIF modules.

PLBV46 Master IPIF Modules

PLBV46 Master Single IPIF Module

The PLBV46 Master Single IPIF module implements the PLB v4.6 baseline feature subset for a master peripheral. Its main target use is for register accesses within user IP. Figure 1-7 shows the block diagram of the PLBV46 Master Single IPIF module.

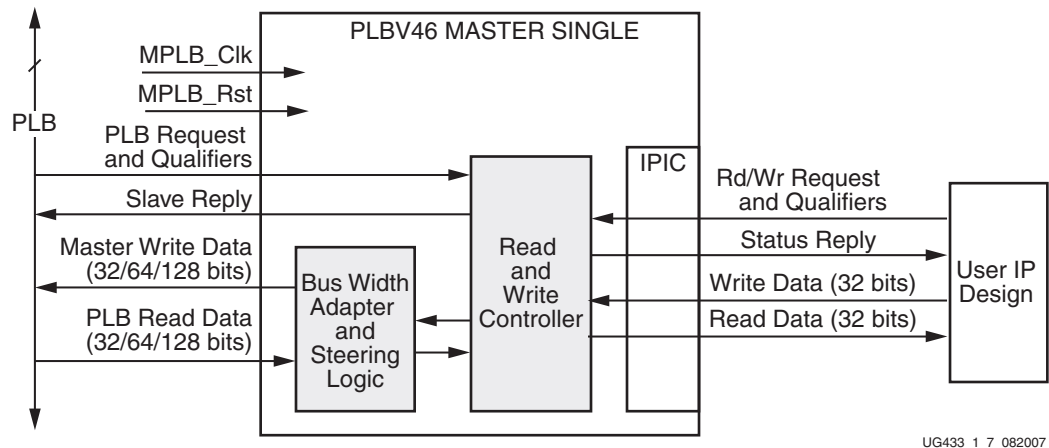


Figure 1-7: PLBV46 Master Single Block Diagram

The PLBV46 Master Single IPIF module supports a 32-bit address bus and a 32-bit data bus. Note that 64-bit and 128-bit data widths are not supported, therefore, a core utilizing this IPIF can only have a data-width of 32-bit. However, cores utilizing this IPIF can connect to PLBV46 busses of 64-bits and 128-bits as required by the IBM 128-bit Processor Local Bus Architectural Specification (v4.6).

Only single read and write data transfers can be initiated by the PLBV46 Master Single IPIF module. Burst transfers and cacheline transfers can not be initiated with this IPIF.

The PLBV46 Master Single IPIC signals used to initiate transfers with the existing OPB and PLBV34 IPIF modules have been replaced with a reduced LocalLink interface. This reduced interface contains source and destination ready signals and read and write data busses. It does not include frame delimiter signals since only single transfers can be initiated.

PLBV46 Master Burst IPIF Module

The PLBV46 Master Burst IPIF module implements the PLB v4.6 performance feature subset for a master peripheral. Its main target use is for high performance, high data throughput masters. Figure 1-8 shows the block diagram of the PLBV46 Master Burst IPIF module.

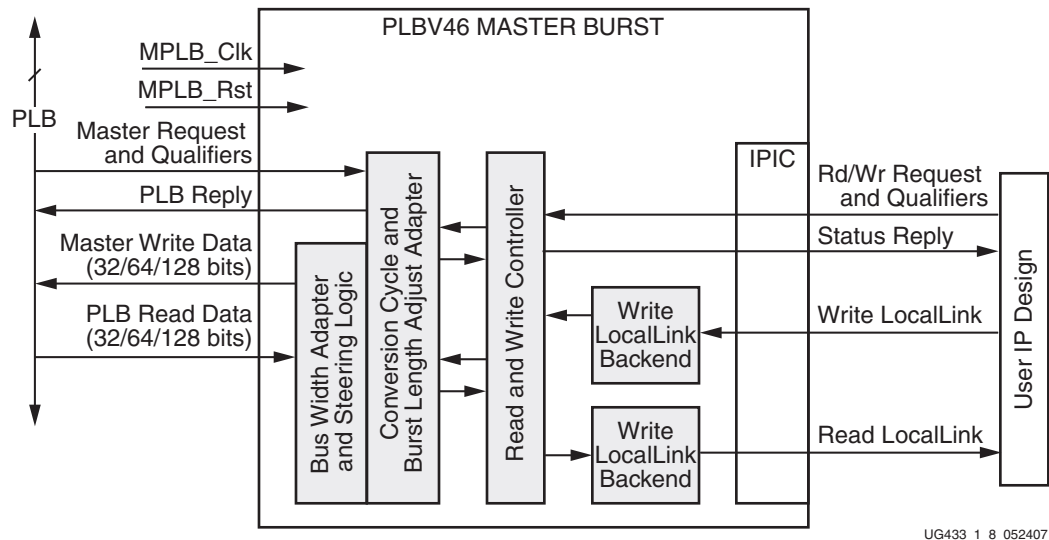


Figure 1-8: PLBV46 Master Burst Block Diagram

The PLBV46 Master Burst IPIF module supports a 32-bit address bus and a 32-bit, 64-bit, or 128-bit data bus. Cores utilizing this IPIF can connect to PLB v4.6 busses of 32-bits, 64-bits and 128-bits as required by the IBM 128-bit Processor Local Bus Architectural Specification (v4.6).

Single read and write data transfers and fixed length burst transfers (up to 16 data beats) can all be initiated by the PLBV46 Master Burst IPIF module to PLB v4.6 slaves of the same or different sizes. With PLBV46 Master Burst, the User IP reads and writes from the PLB Master via the Xilinx LocalLink Interface protocol which is a different protocol than PLBV34/OPB IPIFs.

Existing OPB and PLB v3.4 master/slave cores now require 2 separate IPIFs for the master and slave interface. For example, PLBV46 Slave Single can be used for slave registers to setup master transactions and the PLBV46 Master Burst can be used for the Master. The Master read/write transactions no longer involves the slave attachment. No internal slave transactions are required to support the master transactions. Therefore, the master no longer supplies an IP2IP address.

Indeterminate length bursts are not supported with PLBV46 Master Burst. The User Logic must only generate fixed length bursts of up to 16 data beats. The length is specified in bytes and a multiple of the Native Dwidth/8. In addition, transactions cannot be aborted. With unaligned addresses, the User IP has to issue single data beat requests until address alignment is established. Critical slave registers that control Master operations must be considered since there is no IP2Bus_Busy signal from the slave.

No master services like DMA are included with the PLBV46 Master IPIFs. The user has the option of adding LocalLink interface(s) to the User Logic connecting to SDMA which is described later inside the Migration of DMA Solutions chapter.

Conversion cycles are supported when transferring data to smaller data-width slaves. See section, "[Master/Slave Mixed Data Width Considerations](#)", for more details.

PLBV46 Master IPIF Parameters

Data width parameters should be given special consideration as to the system that this IP will be used in. These parameters are discussed in detail in the Migration of User IP Master/Slave Cores chapter. When these parameters are of mis-matched values, additional logic within the master PLBV46 IPIF may be required in order to support burst run length expansion and conversion cycles. Please refer to the section, “[Master/Slave Mixed Data Width Considerations](#)” of this chapter for more information. Note, however, that this automatic inclusion of the conversion cycle logic and burst run length expansion logic based on the can be turned off via a parameter.

PLBV46 Master IPIC Signal Comparison

The PLBV46 Master IPIC ports are much different from the OPB and PLBV34 IPIC Ports. The PLBV46 Master IPIC now contains a command interface, a read LocalLink interface, and a write LocalLink interface. The command interface is used to setup the transfer and the read and write LocalLink interfaces are used for the actual data transfer. Please refer to the data sheets for the PLBV46 Master IPIF module of interest for more details and descriptions of these signals.

The signals for the IPIC Command Interface are seen in [Table 1-2](#).

Table 1-2: PLBV46 Master Burst and PLBV46 Master Single IPIC Command Interface Signals

Port Name	I/O	Description
IP2Bus_MstRd_Req	I	User logic read request.
IP2Bus_MstWr_Req	I	User logic write request.
IP2Bus_Mst_Addr	I	User logic request byte enables.
IP2Bus_Mst_BE	I	User logic request length in bytes for fixed length burst transfers.
IP2Bus_Mst_Type	I	User logic request type indicator. 0 - single data beat, 1 - fixed length burst.
IP2Bus_Mst_Lock	I	User logic bus lock request.
IP2Bus_Mst_Reset	I	Optional user logic reset request.
Bus2IP_Mst_CmdAck	O	Command acknowledge status.
Bus2IP_Mst_Cmplt	O	Command complete status.
Bus2IP_Mst_Error	O	Command error status.
Bus2IP_Mst_Rearbitrate	O	User logic should ignore this signal
Bus2IP_Mst_Cmd_Timeout	O	Command timeout status.

Some of the command interface signals have similar functionality to the existing OPB and PLBV34 IPIC signal as shown in [Figure 1-9](#) and [Figure 1-10](#).

OPB	PLBV34	PLBV46
Bus2IP_MstWrAck	Bus2IP_MstWrAck	
Bus2IP_MstRdAck	Bus2IP_MstRdAck	
Bus2IP_MstRetry	Bus2IP_MstRetry	
Bus2IP_MstError	Bus2IP_MstError	Bus2IP_Mst_Error
Bus2IP_MstTimeout	Bus2IP_MstTimeout	Bus2IP_Mst_Timeout
Bus2IP_MstLastAck	Bus2IP_MstLastAck	
Bus2IP_IPMstTrans	Bus2IP_IPMstTrans	
		Bus2IP_Mst_CmdAck
		Bus2IP_Mst_Cmplt
		Bus2IP_Mst_Rearbitrate

DS443_1_9_082007

Figure 1-9: Comparison of Master IPIC Signals (Bus2IP)

OPB	PLBV34	PLBV46
IP2Bus_Addr	IP2Bus_Addr	IP2Bus_Mst_Addr
IP2Bus_MstBE	IP2Bus_MstBE	IP2Bus_Mst_BE
IP2IP_Addr	IP2IP_Addr	
IP2Bus_MstWrReq	IP2Bus_MstWrReq	IP2Bus_MstRd_Req
IP2Bus_MstRdReq	IP2Bus_MstRdReq	IP2Bus_MstWr_Req
IP2Bus_MstBurst	IP2Bus_MstBurst	IP2Bus_Mst_Type
IP2Bus_MstBusLock	IP2Bus_MstBusLock	IP2Bus_Mst_Lock
	IP2Bus_MstNum	
		IP2Bus_Mst_Length
		IP2Bus_Mst_Reset

DS443_1_10_082007

Figure 1-10: Comparison of Master IPIC Signals (IP2Bus)

For the actual transfer of data, the LocalLink interfaces are used. Since the PLBV46 Master Single IPIF module can only initiate single transactions and can not initiate burst transactions, this interface signal set is reduced from the signal set for the PLBV46 Master Burst IPIF module.

The PLBV46 Master Burst IPIC Read LocalLink Interface Signals are seen in [Table 1-3](#) and the IPIC Write LocalLink Interface Signals are seen [Table 1-4](#). The PLBV46 Master Single IPIC Read Data Interface and IPIC Write Data Interface are shown in [Table 1-5](#) and [Table 1-6](#).

Note: For a write master transaction, the User Logic is the source and the Master is the destination. For a read master transaction, the User Logic is the destination and the Master is the source.

Table 1-3: PLBV46 Master Burst IPIC Read LocalLink Interface Signals

Port Name	I/O	Description
Bus2IP_MstRd_d	O	Read data output to user logic.
Bus2IP_MstRd_REM	O	LocalLink remainder indicator.
Bus2IP_MstRd_sof_n	O	Active low signal indicating the starting data beat of a read LocalLink transfer.
Bus2IP_MstRd_eof_n	O	Active low signal indicated the ending data beat of a read LocalLink transfer.
Bus2IP_MstRd_src_rdy_n	O	Active low signal indicating that the data value asserted on the Bus2IP_MstRd_d bus is valid.
Bus2IP_MstRd_src_dsc_n	O	Active low signal that the Source needs to discontinue the transfer.
IP2Bus_MstRd_dst_rdy_n	I	Active low signal indicating that the data value asserted on Bus2IP_MstRd_d Bus is being asserted by the destination.
IP2Bus_MstRd_dst_dsc_n	I	Active low signal indicating that the Destination (User logic) needs to discontinue the transfer.

Table 1-4: PLBV46 Master Burst IPIC Write LocalLink Interface Signals

Port Name	I/O	Description
IP2Bus_MstWr_d	I	Write data input from User logic.
IP2Bus_MstWr_REM	I	LocalLink remainder indicators.
IP2Bus_MstWr_sof_n	I	Active low signal indicating the starting data beat of a write LocalLink transfer.
IP2Bus_MstWr_src_rdy_n	I	Active low signal indicating that the data value asserted on IP2Bus_MstWr_d Bus is valid.
IP2Bus_MstWr_src_dsc_n	I	Active low signal indicating that the data value asserted on IP2Bus_MstWr_d bus is being asserted by the Destination.
Bus2IP_MstWr_dst_rdy_n	O	Active low signal indicating that the data value asserted on IP2Bus_MstWr_d bus is being asserted by the Destination.
Bus2IP_MstWr_dst_dsc_n	O	Active low signal indicating that the Write LocalLink Destination(Master) needs to discontinue the transfer.

Table 1-5: PLBV46 Master Single IPIC Read Data Interface Signals

Port Name	I/O	Description
Bus2IP_MstRd_d	O	Read data output to User Logic.
Bus2IP_MstRd_src_rdy_n	O	Active low signal indicating that the data value asserted on the Bus2IP_MstRd_d Bus is valid.

Table 1-6: PLBV46 Master Single IPIC Write Data Interface Signals

Port Name	I/O	Description
IP2Bus_MstWr_d	I	Write data input from the User Logic.
IP2Bus_MstWr_dst_rdy_n	O	Active low signal indicating that the data value asserted on the IP2Bus_MstWr_d Bus has been transferred on the PLB.

Example state machines used in the User Logic to interface to these LocalLink interfaces are created by the Create/Import IP Wizard.

Master/Slave Mixed Data Width Considerations

Since a 128-bit data width masters can exist and most PLB v4.6 slaves are 32-bit data width, it is important to understand the system implications of having mixed data widths masters and slaves.

Burst Run Length

The data width of a burst transaction is determined by the value of the PLB_Size signals. The length of a burst transaction is determined by the value of the PLB_BE signals. The width of the PLB_Size vector is the same for masters and slaves regardless of their data widths. However, since the PLB_BE vector at each peripheral is sized as the data width of the peripheral / 8 since for single transfer types these signals indicate the valid data byte lane. Therefore, the size of the PLB_BE vector can be different at peripherals of different data widths. For this reason, the Xilinx PLB v4.6 interconnect limits the length of a burst to 16 data beats so that this length can always accurately represented on the PLB_BE vector for all devices. (The smallest data width allowed for peripherals in the system is 32-bits which dictates a PLB_BE vector of 4 bits which adequately represents a fixed length burst of 16 data beats).

Note, however, that the PLB_BE vector represents the number of data beats as indicated by the PLB_Size vector. For a 128-bit data width master, the value of the PLB_BE vector can represent the length of burst of 128-bit wide data beats. If this transaction is received by a 32-bit data width slave, the PLB_BE vector value no longer accurately represents the amount of data in the transaction.

Therefore, when a slave is connected to a wider data-width master, the slave is required to dynamically adjust their burst run length. Likewise, when a master is connected to a narrower data-width slave, the master is required to dynamically adjust the burst length if the size of the fixed length burst exceeds the native data width of the target slave.

Slaves are also required to adjust the number of data beats needed for cacheline transfers when the transaction involves mixed slave/master data widths.

Burst Run Length Example:

An example of a 128-bit PLB v4.6 master requesting a fixed length burst of 16 double-words to a 32-bit data-width slave is described below:

- ◆ 128-bit PLB v4.6 Master requests a 16 beat burst of double words of a 32-bit Slave
 - PLB_Size = "1011", PLB_BE = "1111"
- ◆ 2 words per double word * 16 data beats = 32 words being requested
- ◆ 32 data beats required from Slave, but PLB_BE only indicates 16
- ◆ Slave must adjust burst run length accordingly to return 32 words
 - Can't use PLB_BE value to determine burst run length
- ◆ Master must also adjust burst run length to know how many data acknowledges needed to complete the burst transaction so that the burst signal is asserted for the proper number of clocks

Conversion Cycles

Masters are encouraged to support Conversion cycles (master accessing narrower data width slaves) if the requested byte enables can cross the native data width alignment of the target slave. When slaves are connected to a wider data-width master or a wider data-width PLB v4.6 interconnect, the slaves must implement a mux which based on the values of the lower address bits, selects the appropriate slice of the larger PLB_BE vector to assign to the PLB_BE vector of the size associated with the slave's data width. This is shown for a 32-bit data-width slave connecting to a 128-bit data-width PLB v4.6 in [Figure 1-11](#) with the exception that the byte enable mux is shown external to the slave in this figure and in the Xilinx PLB v4.6 implementation, this mux is part of the slave device.

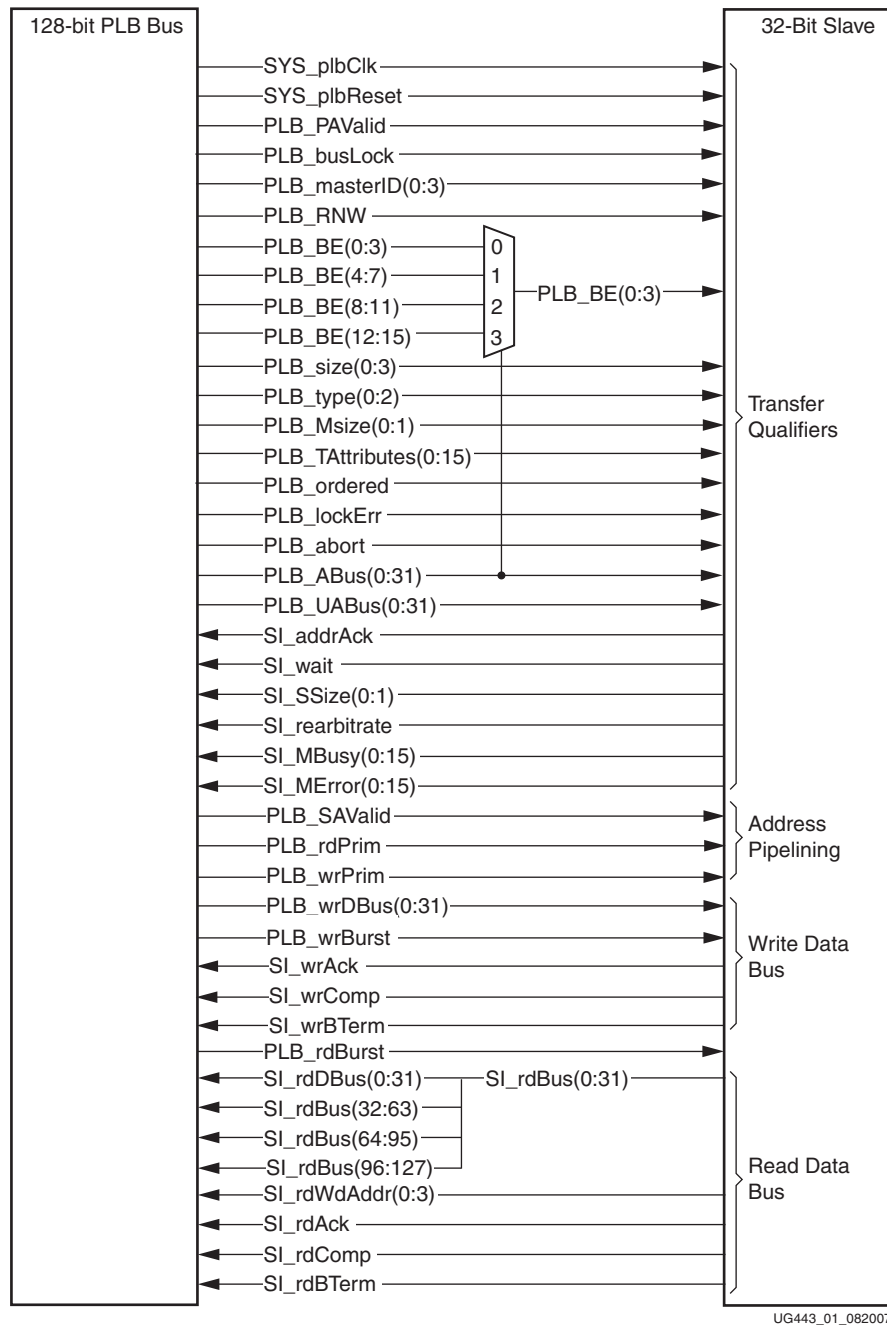


Figure 1-11: Byte Enable Muxing 32-bit Data-width Slave, 128-bit Data-width PLBV46 bus

Conversion Cycle Example:

An example of a 128-bit data-width master performing conversion cycles to a 32-bit data-width slave is described below:

- ◆ 128-bit Master requesting a single read of two bytes from address 0x07 from a 32-bit slave:
 - PLB_ABus = 0x07, PLB_BE = 0x0180 (0000 0001 1000 0000)
- ◆ BEs cross the boundary of the 32-bit slave
- ◆ Slave is required to mux BEs based on PLB_ABus(28:29)
- ◆ Slave BEs = 0001, Slave returns 32-bits from address 7
- ◆ Master detects slave is 32-bits, generates conversion cycle:
 - PLB_ABus = 0x08, PLB_BE = 0x0080 (0000 0000 1000 0000)
- ◆ Slave BEs = 1000, Slave returns 32-bits from address 8

Read Data Bus Steering

When a slave is accessed by a smaller data-width master for a read transaction, the slave must properly place the data on the correct byte lanes of the read data bus so that the master can access the read data. This is called read data bus steering. Figure 1-12 shows the connections between a 128-bit data-width slave and a 32-bit data-width master.

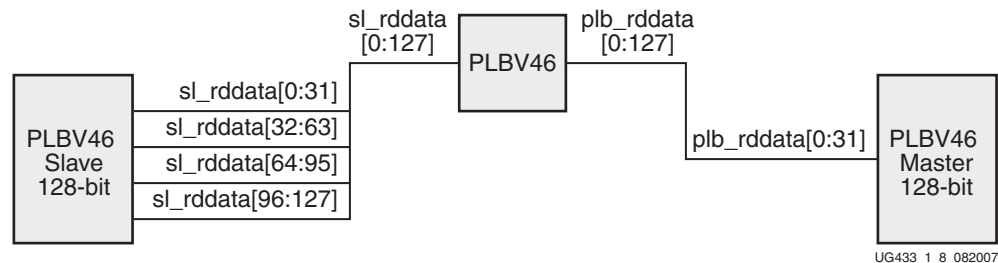


Figure 1-12: 128-bit Data-width Slave Connected to a 32-bit Data-width Master

Read Data Bus Steering Example:

An example of a 32-bit data-width master requesting a read from a 128-bit data-width slave is described below:

- ◆ 32-bit master requesting a read from a 128-bit slave from address 0x0C
- ◆ Data from 0x0C would appear on Sl_RdData(96:127)
- ◆ Due to 32-bit master size, slave steers data to Sl_RdData(0:31)

Mixed Master/Slave Data-width Support in Xilinx IP

PLBV46 Master Burst and PLBV46 Slave Burst IP will also support burst run length adjustment and data steering and mirroring. These cores can also support conversion cycles.

PPC405 - PPC440 System Migration

Introduction

This chapter describes the migration process of a Xilinx embedded processor system from a PPC405 OPB/PLB v3.4 system to a PowerPC™ 440 with PLB v4.6/XPS system.

The following high-level migration steps will be discussed:

- PPC440 processor block
- PLB v4.6 bus instances are added to the system where OPB/PLB v3.4 buses are removed
- Clocking/Reset scheme
- Equivalent XPS cores (from OPB/PLB v3.4) are connected to PLB v4.6 instances
- Ethernet solution
- Existing memory controllers are migrated to either the PPC440MC DDR2 or MPMC

The original system is built for a Virtex™-4FX PowerPC 405 system similar to a system for the ML410 board and the migrated system is built for a Virtex-5FXT PowerPC 440 system with a generic board.

System Migration Methodology

System Hardware Migration

The method that is recommended for doing the system hardware migration is outlined below.

- Migrate the processor from the PPC405 to the PowerPC 440
- Separate out the main memory, slave and master/slave cores of the existing PPC405 system
- Migrate PLB DDR2 or MCH OPB DDR2 memory controllers to the equivalent PPC440MC DDR2 memory controller
 - ◆ If an existing SDRAM or DDR memory controller is inside the system, the MPMC memory controller is used
- Master cores
 - ◆ These are connected to either the SPLB0 or SPLB1 or MPLB PLB v4.6 bus ports on the processor block depending on functionality of the master
 - ◆ The maximum number of masters connected to SPLB0 or SPLB1 is 4 each (8 total)

- Slave cores
 - ◆ These are connected to the MPLB bus port on the processor block
 - ◆ The maximum number of slaves connected to MPLB is 16
- Utilize the DMA controller or the XPS Central DMA, as appropriate depending on the system
- Determine clocking requirements for the PowerPC 440 system and setup the PLL and clock generator
- Determine the reset requirements and the interrupt connections

System Software Migration

The basic considerations for migrating the user applications from the PPC405 to the PowerPC 440 system are listed below. A more detailed section covering software migration is discussed later in this chapter.

- Set software platform settings correctly
- Review cache requirements and cache sizes if necessary
- Set compiler options appropriately for each SW application

Existing PPC405 OPB/PLB v3.4 System

For reference, an example PPC405 system is shown in [Figure 2-1](#). To highlight typical steps in the migration process, this system is shown with some of the common cores found in an embedded system.

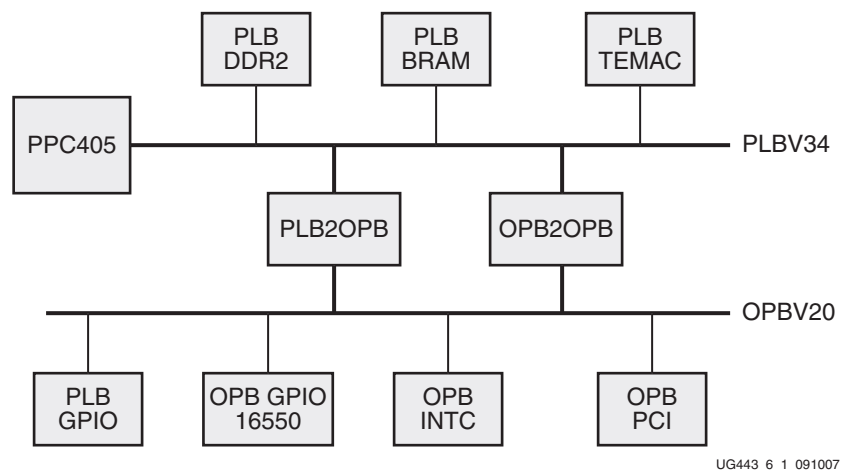


Figure 2-1: PPC405 Example System

PowerPC 440/XPS System

The migrated PowerPC 440 system is shown in [Figure 2-2](#).

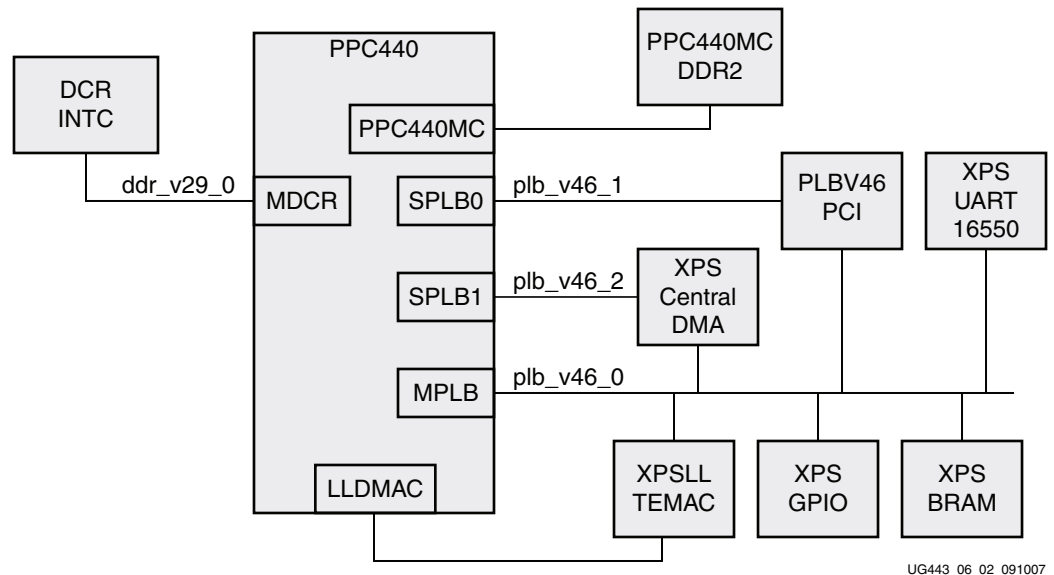


Figure 2-2: **PowerPC 440 Migrated System**

Overview of System Migration

Processor Migration

Within the EDK project, the PowerPC 440 Virtex-5 is added to the system.

The PPC440 processor block allows the system to connect PLB Masters, PLB Slaves, DCR Master/Slaves, DMA devices through the crossbar. PLB Slave peripherals connect to the MPLB port and PLB Masters connect to the SPLB0 and SPLB1 ports on the processor block. Either the PPC440MC DDR2 or MPMC with the MIB PIM can connect to the PPC440MC on the processor block.

The MPLB and PPC440MC bus interfaces look like a slave connections to the PowerPC 440 processor block.

SPLB0 and SPLB1 bus interfaces look like master connections to the PowerPC 440 processor block.

The four LocalLink/DMA interfaces provide DMA to the MPLB and/or PPC440MC inside the system. DMA operations are accomplished through the LocalLink protocol.

Refer to the *Virtex-5 Embedded Processor Block for PowerPC 440 Designs Reference Guide* for more details about the PowerPC 440.

Addressing Inside the PowerPC 440 Processor Block

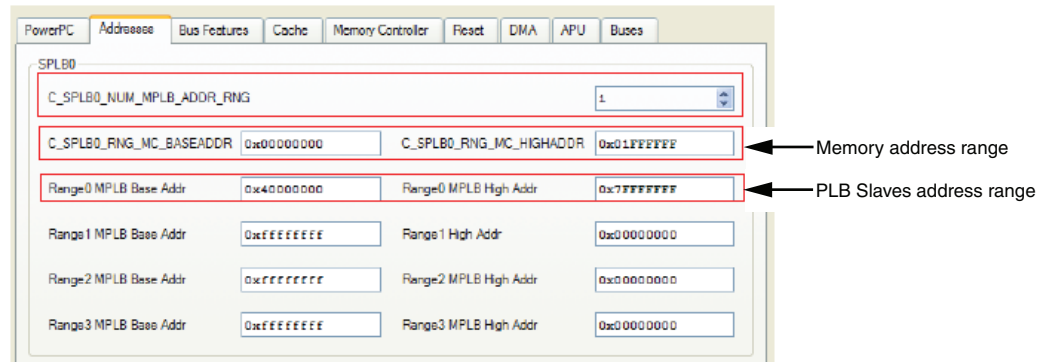
There are two slave ports on the PowerPC 440 processor block, the Memory Controller (PPC440MC) port and the MPLB port that are configured through the EDK PPC440 wrapper.

There are two master ports, SPLB0 and SPLB1. Master PLB v4.6/XPS devices should be connected to either one of these ports.

The memory address ranges for each master port to the slave ports has to be specified. For SPLB0 and SPLB1:

- Set the base address and high address for the PPC440MC
- Set the base address and high address for the MPLB

Set the address for each SPLB0 and SPLB1 through the EDK GUI as shown in [Figure 2-3](#).



DS443_6_3_090607

Figure 2-3: Setting Address Ranges for SPLB0 and MC

In this example, the MPLB is configured for only one address range. Masters connected to SPLB0 and SPLB1 will see one address range for memory and one address range, in this system, for slaves:

- Memory, via addresses 0x0000_0000 - 0x0FFF_FFFF
- Other PLBV46 slaves, via addresses 0x4000_0000 - 0x7FFF_FFFF

The user is allowed to configure up to four independent ranges for the slave peripherals connected to the MPLB port. This is beneficial during migration when main memory is in conflict with MPLB addresses. An example of this is when PLB DDR2 is migrated to PPC440MC DDR2. In this case, the address space of PLB DDR2 would be in the MPLB address space if all peripherals were mapped to one address space. The address ranges need to set identically for SPLB0 and/or SPLB1.

Figure 2-4 shows an address mapping example for migrating from the PPC405 to the PowerPC 440. In this example, the PLB DDR2 memory controller is replaced by the PPC440MC DDR2 so that is the address range for the MC. The PLB BRAM Controller is replaced by the XPS BRAM Controller so its address is mapped to the MPLB range 0. The PLB GPIO is replaced with the XPS GPIO so its address is mapped to MPLB range 1.

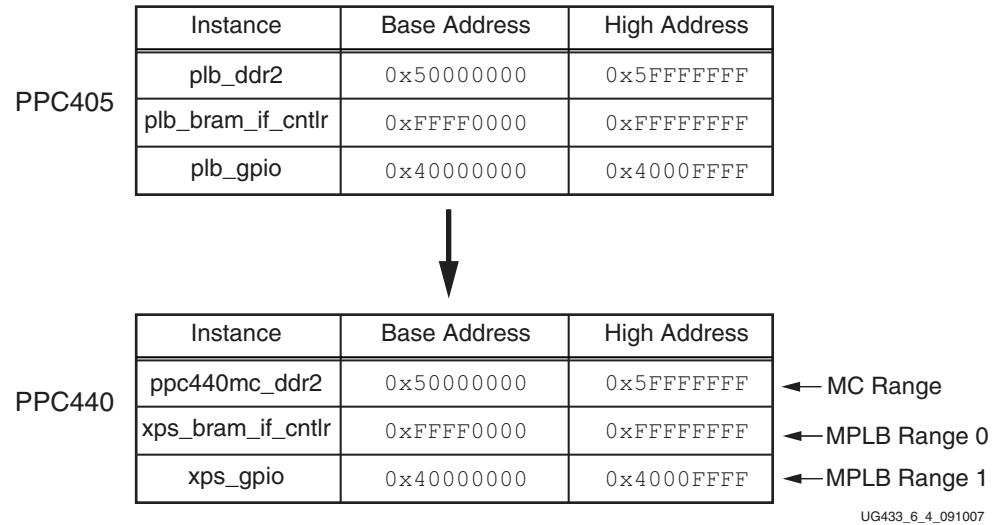


Figure 2-4: Address Mapping Example

Interconnect

The PLB v3.4 and OPB instances are deleted and PLB v4.6 instances are added for the MPLB, SPLB0, and SPLB1 as needed. In addition, the PLB2OPB and OPB2PLB bridges are removed.

Connecting Peripherals to MPLB, SPLB0, SPLB1

A PLB v4.6 bus instance is used on the MPLB to connect slave peripherals to the system. PLB v4.6 bus instances are used for SPLB0 and SPLB1. Two master peripherals could be on the same bus but there are pros and cons for each scenario.

- Two master PLBs (using SPLB0 and SPLB1)
 - ◆ Higher performance but additional resources
- One master PLB (using either SPLB0 or SPLB1)
 - ◆ Lower resource utilization but lower performance
 - ◆ Would need arbitration logic for multiple masters

It is also possible to connect both masters to the MPLB (slave) bus to realize an even lower resource utilization, but it would adversely affect the performance of the system and would need arbitration logic. In this case, masters on the MPLB could only initiate transactions to slaves on the MPLB and not to the PPC440MC port.

PLB v4.6 DWIDTH Settings

Platgen sets the DWIDTH parameter for each bus instance and all the peripherals connected to each bus to be the maximum value of the NATIVE_DWIDTH parameter among all the peripherals connected to the bus. Per the IBM 128-bit Processor Local Bus Architectural Specification (v4.6), the PLB v4.6 bus must be at least as wide as the widest

connected master/slave interface. All narrower masters/slaves adapt their connections to connect to the wider bus using bit mirroring and (for slaves) ByteEnable muxing. Masters would use conversion cycles. The NATIVE_DWIDTH for all PLB v4.6 ports on the PowerPC 440 is 128 bits and is invariant. This will always cause platen to set the DWIDTH of all buses connected to any of these PowerPC 440 ports to 128 bits. During operation, any of the PowerPC 440 PLB v4.6 interfaces will accept all transactions from, and issue compatible transactions to, other peripherals of any NATIVE_DWIDTH (32, 64, or 128 bit).

Memory Controller Migration

The DDR2 memory controller has been replaced with the PPC440MC DDR2 memory controller which connects directly to the Memory Interface Block (MIB) on the PPC440 processor block. By having the dedicated port for memory transfers, the PLB v4.6 bus is free of direct processor-to-memory transactions.

For DDR or SDRAM memory controller system migration, MPMC is connected to the MIB through a port configured for the MIB Port Interface Module (PIM).

Interrupt Controller

The OPB Interrupt Controller has been replaced with the DCR Interrupt Controller (INTC) which connects directly to the DCR port on the processor block. The XPS INTC core could have also been used which connects to the MPLB PLB v4.6 bus instance.

For the example migrated system shown, the DCR INTC was chosen to demonstrate the use of the DCR port on the processor block. A DCR bus will have to be added into the system to connect the DCR INTC to the processor block.

There are two DCR ports on the processor block, the SDCR port which is used to connect master devices on the DCR bus, and the MDCR which is used to connect slave devices. The DCR INTC connects to the MDCR port.

The address ranges for the DCR need to be set up in both the PowerPC 440 configuration and the DCR INTC configuration.

The address range set in the PowerPC 440 configuration sets the address range for the DCR registers inside the processor block. The address range set for the DCR INTC configuration sets the address range for the peripheral.

XPS Peripherals

This section covers the general migration from an OPB/PLB v3.4 inside the system to a XPS core. This is applicable to slave and master/slave cores.

1. The former PLB v3.4 or OPB core is deleted.
2. The XPS core is added to the system.
3. Parameters and ports are connected in the same manner as the PLB v3.4 or OPB equivalent.

The slave or master/slave interfaces are connected to the PLB v4.6 shared bus instance

In this case, The XPS BRAM and XPS UART16550 cores replace the PLB BRAM and OPB UART16550 cores, respectively. These slave devices are connected to the MPLB port PLB v4.6 bus instance which connects all XPS slave cores in the system.

The OPB PCI core is replaced by the PLBV46 PCI core. This core has a connection to the slave PLB v4.6 bus for configuring the core and also has a master connection. The master connection can be connected to SPLB0 or SPLB1 (master) buses to enhance performance.

Another consideration for the PLBV46 PCI core is that, unlike the OPB PCI core, it does not have an internal SGDMA. Therefore, to handle SGDMA operations for the PCI, an external DMA Controller must be added into the system. The XPS Central DMA core is used to provide DMA to the system.

The XPS Central DMA has a connection to the slave PLB v4.6 bus for configuring the slave registers for DMA operations and a master connection. The master connection can be connected to one of the SPLB0 or SPLB1 (master) buses to enhance performance.

System Clocking

The PowerPC 440 processor block requires PLL outputs to drive some of the clocks. The PLL adds delays to certain clocks to match internal clock delays of the PowerPC 440 processor block. Setting the value for C_CLKOUTn_DESKEW_ADJUST to PPC will automatically add the proper delay for the outputs that drive MPLB, SPLB0, SPLB1, MC (MIB), and APU clocks.

The PLL module has six available outputs. The output frequencies of each of the clock outputs can be set independently as multiples of the input clock. The parameters C_CLKFBOUT_MULT and C_CLKOUTn_DIVIDE set the output frequency for each clock output.

The PowerPC 440 Processor Block clocking structure is shown in Figure 2-5. The PLL delays are shown going to the PLB I/F (PLB Clock), the MIB I/F (MI Clock), and the APU (FCM Clock).

Also shown in Figure 2-5 is an example clocking ratio scheme for the processor block clocks.

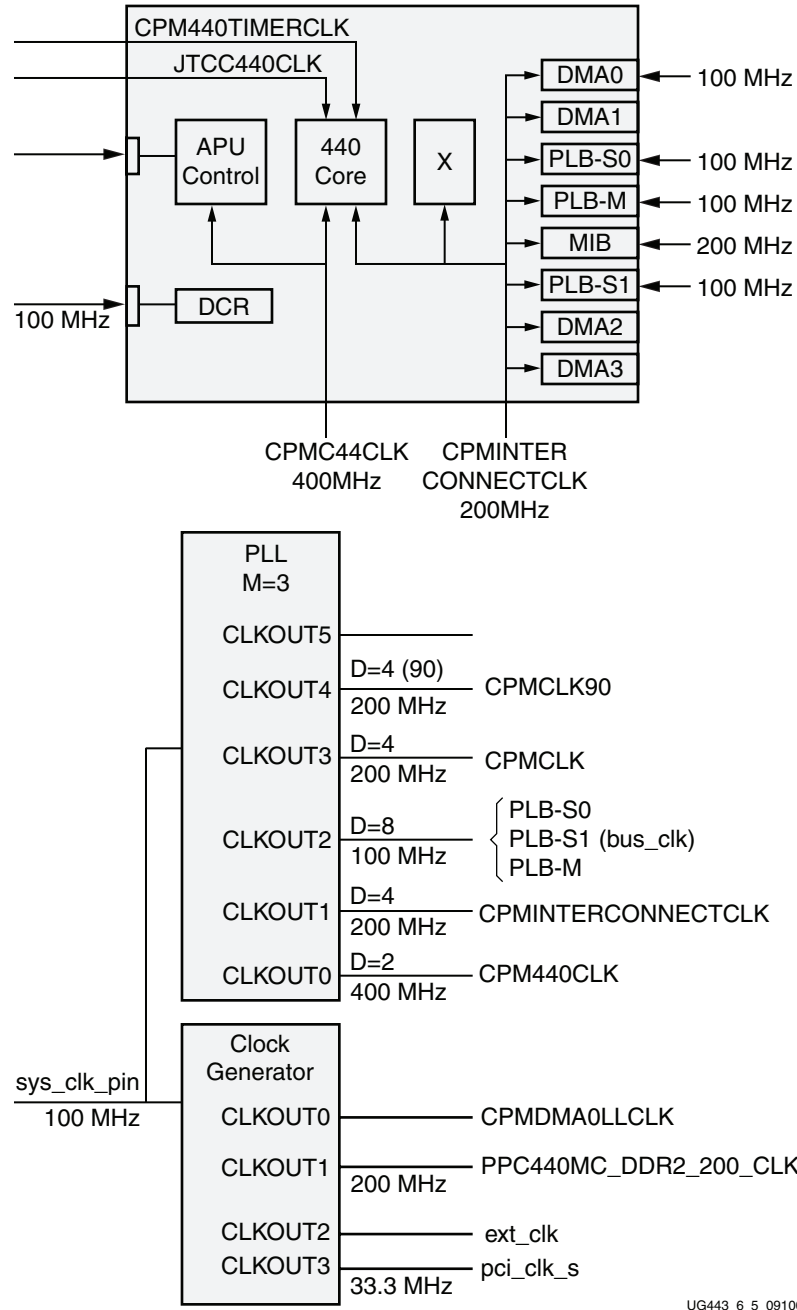


Figure 2-5: Processor Block Clock Diagram

Reset Structure

The PowerPC 440 reset structure is similar to a PPC405 system. The `proc_sys_reset` module is using the latest version. The PPC440 processor block has the same reset bus interface as the PPC405 system that connects to the `proc_sys_reset` module.

It is recommended to use separate resets to each bus from the `Bus_Struct_Reset` output of the `proc_sys_reset` module. For master and slave peripherals connected to PLB v4.6 instances, the peripherals get vectorized resets for each master and slave connection from the bus core.

Ethernet Controller Migration

The migration of the PLB TEMAC to the XPS LL TEMAC controller varies depending on the features configured for the PLB TEMAC core.

If the PLB TEMAC core in the PPC405 PLB v3.4 system was configured to include the Scatter-gather DMA feature, the migrated system will require a DMA Controller with Scatter-gather DMA capability to be added. The DMA controller on the processor block provides Scatter-gather DMA capability to either the MPLB or PPC440MC ports. The XPS LL TEMAC connects to this DMA controller via the LocalLink port.

If the PLB TEMAC core in the PPC405 PLB v3.4 system was configured to use FIFO direct mode, the migrated system will require the addition of the XPS LL FIFO core. The XPS LL TEMAC connects to the XPS LL FIFO via the LocalLink port. The FIFO data is then accessed through the PLB v4.6 slave connection on the XPS LL FIFO.

XPS LL TEMAC instantiates either a hard TEMAC or a soft TEMAC into the core. For a V4FX/V5FXT system, XPS LL TEMAC will instantiate the hard TEMAC. XPS LL TEMAC has two hard TEMACs available inside the core.

Migration of the JTAG Controller

An updated version of the JTAG Controller has been developed for use with the PowerPC 440 processor block. It is `jtagppc_cntlr` version 2.01.a This will directly replace the original PPC405 JTAG Controller.

System Migration Inside an EDK System

There are two basic methods for migrating from a PPC405 system to the PowerPC 440 system. One way is to start an EDK project from scratch and add in processor and the equivalent buses and peripherals. The other method is to migrate by starting with the original system and replacing the processor, buses and peripherals and making the appropriate internal/external connections and parameter changes. The following general steps can be a reference for doing either migration method.

Before migrating to the PowerPC 440 processor inside the EDK project, the Virtex-5FXT device can be selected through the EDK Project Options GUI. This allows the user to pick the specific device size, package, and speed grade.

In addition, migrating from a Virtex-4 device to a Virtex-5 device requires modifications to be made to the UCF file for LOC constraints if using an existing system as a starting place.

Migrating Buses/Processor

Migrating the Bus

Adding/Removing Bus Instances

Remove the `opb`, `plb`, `plb2opb` and `opb2plb` instances (Delete instance and its internal ports) inside the System Assembly View/Bus Interfaces.

In addition, add 3 PLB v4.6 instances by expanding the Bus Bridge tree node. Right click on **Processor Local Bus (PLB) 4.6** and select **Add IP**. The buses are for the MPLB, SPLB0, and SPLB1. The instances created are `plbv46_0`, `plbv46_1`, and `plb_v46_2`.

Add the DCR bus by expanding the Bus Bridge tree node. Right click on **Device Control Register (DCR) Bus 2.9** and select **Add IP**. This creates the `dcr_v29_0` instance.

Migrating the Processor

Remove the former `ppc405_0` instance by right clicking on `ppc405_0` inside the **System Assembly View/Bus Interfaces** and clicking on **Delete Instance....** (Delete instance and its internal ports).

Add the PowerPC 440 Virtex-5 to the system by expanding the Processor or the Global Peripheral Repository tree node inside the IP Catalog tab. Right click on **PowerPC 440 Virtex-5** and click on **Add IP**. This creates the `ppc440_virtex5_0` instance.

Setting Parameters

Right click on `ppc440_virtex5_0` inside the System Assembly View and select **Configure IP....**

Inside the **Addresses** tab, set the addresses for the DCR, SPLB0 and SPLB1.

The Base and High Address for the Memory Controller (MemCon) is automatically set by the tools.

Set the Internal DCR Register Base Address and Internal DCR Register High Address to `0b0000000000` and `0b0011111111`, respectively. This sets the address range for the DCR registers inside the processor block.

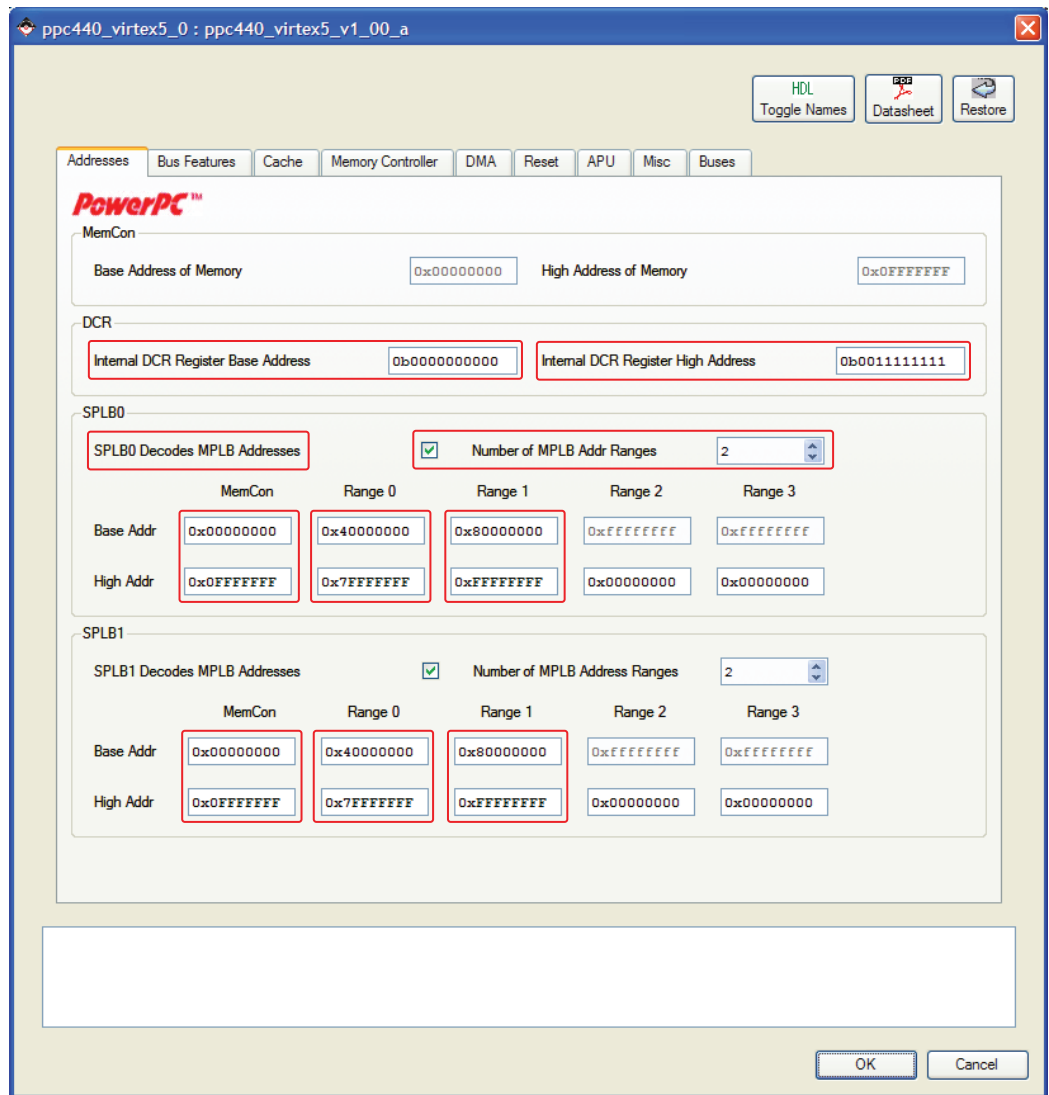
SPLB0 and SPLB1 address parameters are set the same.

Check SPLB0 Decodes, MPLB Addresses, and SPLB1 Decodes MPLB Addresses. This allows masters on either SPLB0 and SPLB1 to access slave peripherals inside the MPLB address range. If unchecked, masters could only access address range inside the PPC440MC.

Set Number of MPLB Addr Ranges for both SPLB0 and SPLB1 to **2**.

Set MemCon Base Addr and High Addr to `0x00000000` and `0x0FFFFFFF`. Set Range0 Base Addr and High Addr to `0x40000000` and `0x7FFFFFFF`. Set Range 1 Base Addr and High Addr to `0x80000000` and `0xFFFFFFFF`. Slave peripherals on the MPLB should have an address range between `0x40000000` and `0xFFFFFFFF`. This applies to both SPLB0 and SPLB1.

The above configuration is shown in Figure 2-6.



UG443_6_6_091007

Figure 2-6: PowerPC 440 Addresses Tab

The control and conflict DCR registers for the PPC440MC memory controller are set in the Memory Controller tab.

The memory controller will be added later in the chapter with the following settings. The memory controller has a 64-bit data width and is configured for 13 bits for row, 10 bits for column, and 2 bits for bank. In addition, the address offset is $\log_2(\text{data_width}/8)$ which is 3 in this case. This information to set the address mapping, after which the Row Conflict Mask and Bank Conflict Mask registers are set.

The address mapping structure for the memory is:

[Bank Address][Row Address][Column Address] [Address Offset]

The setup for this example is shown in [Figure 2-7](#).

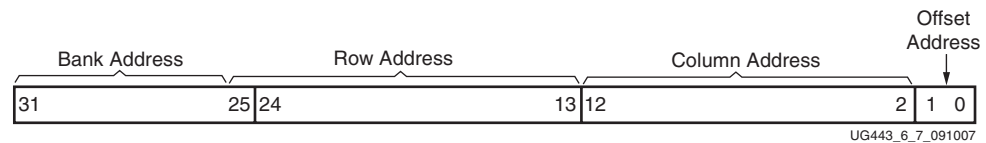
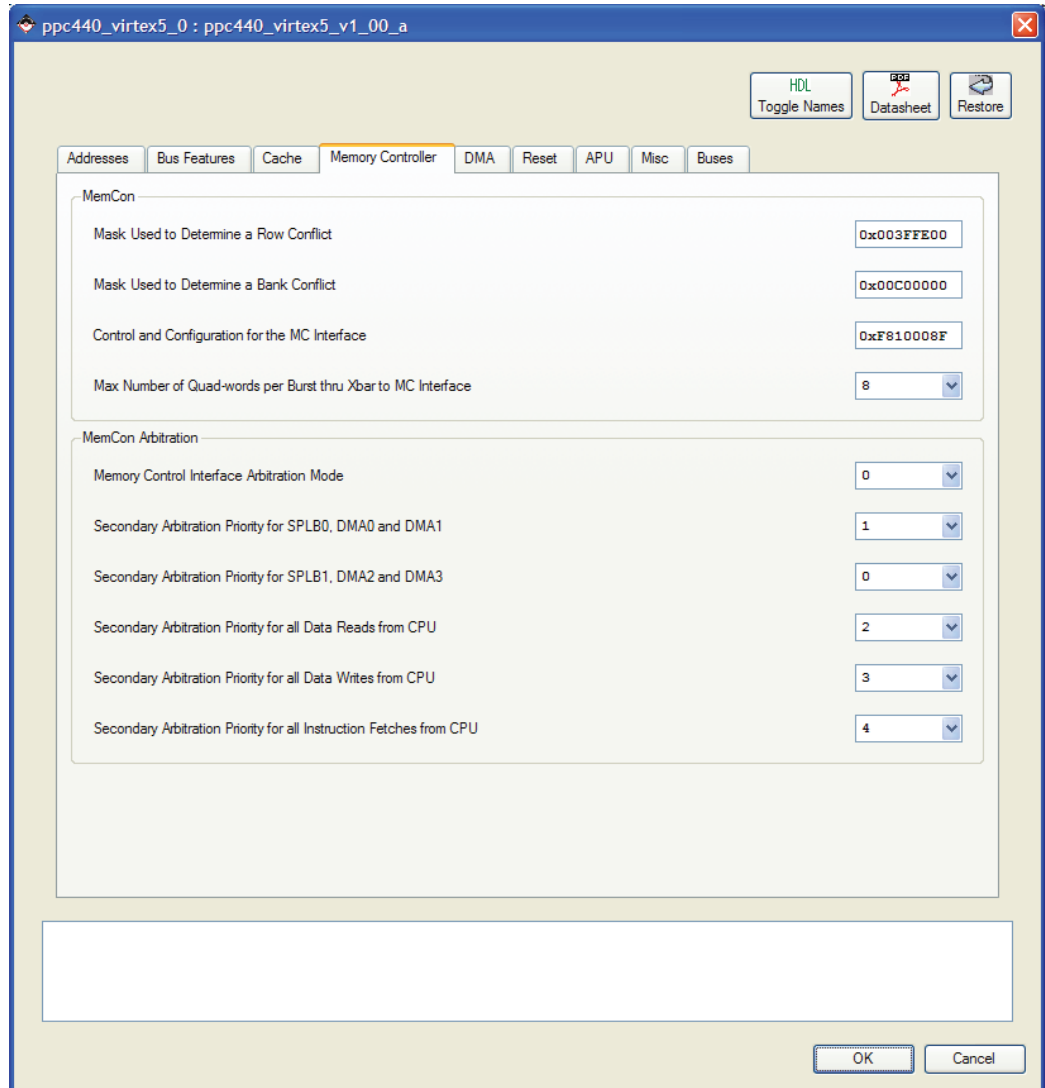


Figure 2-7: Example Address Map

Set Mask Used to Determine a Row Conflict to **0x003FFE00** and Mask Used to Determine a Bank Conflict to **0x00C00000**. These values are derived from placing 1s (ones) inside the bit locations in [Figure 2-7](#) for the row and bank bits. Row bits are between bits 25 to 13 and bank bits are between bits 27 to 26.

Set the Control and Configuration for the MC Interface to **0xF810008F**. To review and set this register, see the Refer to the *Virtex-5 Embedded Processor Block for PowerPC 440 Designs Reference Guide*.

The above parameters are shown in [Figure 2-8](#).



UG443_6_8_091007

Figure 2-8: PowerPC 440 Memory Controller Tab

In the DMA tab, set the Number of DMA Channel to **1**. The DMA Channel is used for the LocalLink interface on the XPS LL TEMAC.

Setting Bus Interfaces

In the Bus Interfaces Tab, expand the ppc440_virtex5_0 instance, make the selections shown in [Figure 2-9](#).

Select the following bus interfaces:

- MDCR bus connection is connected to dcr_v29_0
- JTAGPPC bus connection is connected to jtagppc_0_JTAGPPC0
- MPLB bus connection is connected to plb_v46_0
- SPLB0 bus connection is connected to plb_v46_1
- SPLB1 bus connection is connected to plb_v46_2

Name	Bus Connection	IP Type	IP Version
ppc440_virtex5_0		ppc440_virtex5	1.00.a
...MFCM	ppc440_virtex5_0_MFCM		
...MFCB	No Connection		
...SDCR	No Connection		
...LLDMA3	No Connection		
...LLDMA2	No Connection		
...LLDMA1	No Connection		
...RESETPPC	No Connection		
...LLDMA0	No Connection		
...MDCR	dcr_v29_0		
...JTAGPPC	jtagppc_cntrl_0_JTAGPPC0		
...PPC440MC	ppc440_virtex5_0_PPC440MC		
...MPLB	plb_v46_0		
...SPLB1	plb_v46_2		
...SPLB0	plb_v46_1		
dcr_v29_0		dcr_v29	1.00.a
plb_v46_0		plb_v46	1.00.a
plb_v46_1		plb_v46	1.00.a
plb_v46_2		plb_v46	1.00.a
xps_bram_if_cntrl_1		xps_bram_if_cntrl	1.00.a
xps_bram_if_cntrl_1_bram		bram_block	1.00.a
ddr2_int		xps_gpio	1.00.a
LEDs_8Bit		xps_gpio	1.00.a
dcr_intc_0		dcr_intc	1.00.b
reset_block		proc_sys_reset	2.00.a
pll_module_0		pll_module	1.00.a
PCI32_BRIDGE		plbv46_pci	1.00.a
pci_arbiter_0		pci_arbiter	1.00.a
xps_ll_temac_0		xps_ll_temac	1.00.a
ppc440mc_ddr2_0		ppc440mc_ddr2	1.00.a
xps_uart16550_0		xps_uart16550	1.00.a
jtagppc_cntrl_0		jtagppc_cntrl	2.01.a
xps_central_dma_0		xps_central_dma	1.00.a
clock_generator_0		clock_generator	1.00.a

UG443_6_9_091007

Figure 2-9: PowerPC 440 Bus Interfaces

Migrating The Memory Controller

Remove the DDR_SDRAM_64Mx32 instance by right clicking on **DDR_SDRAM_32Mx64** inside the System Assembly View/Bus Interfaces and clicking on **Delete Instance...**, then click on **Delete instance and its internal ports**.

Add the PPC440MC DDR2 to the system by expanding the Memory and Memory Controller or Global Peripheral Repository 0 tree node inside the IP Catalog tab. Right click on **PPC440MC_DDR2** and click on **Add IP**. This creates the `ppc440_mc_ddr2_0` instance.

See the PPC440MC DDR2 documentation for IOSTANDARDS required for certain pins for the UCF which might be different from the previous memory controller.

Configuring the PPC440MC DDR2 core

Setting Parameters

Right click on **ppc440mc_ddr2_0** in the System Assembly View and select **Configure IP...**

In the All tab, set the following parameters:

- Set `C_DDR_BAWIDTH` to **2**
- Set `C_DDR_DWIDTH` to **64**
- Set `C_DDR_CAWIDTH` to **10**
- Set `C_DDR_DM_WIDTH` to **8**
- Set `C_DQ_BITS` to **8**
- Set `C_DQS_BITS` to **3**
- Set `C_DDR_RAWIDTH` to **13**
- Set `C_MIB_MC_CLOCK_RATIO` to **1**
- Set `C_MEM_BASEADDR` and `C_MEM_HIGHADDR` to **0x00000000** and **0x0FFFFFFF**, respectively
- Set `C_MEM_CLK_PERIOD_PS` to **5000**
- Set `C_NUM_IDELAYCTRL` to **3**
- Set `C_IDELAYCTRL_LOC` to **IDELAYCTRL_X0Y5-IDELAYCTRL_X0Y4-IDELAYCTRL_X0Y3**

PPC440MC DDR2 Bus Interface

In the Bus Interfaces Tab, expand the `ppc440mc_ddr2_0` instance. The PPC440MC bus interface is connected to `ppc440_virtex5_0_PPC440MC` as shown in Figure 2-10.

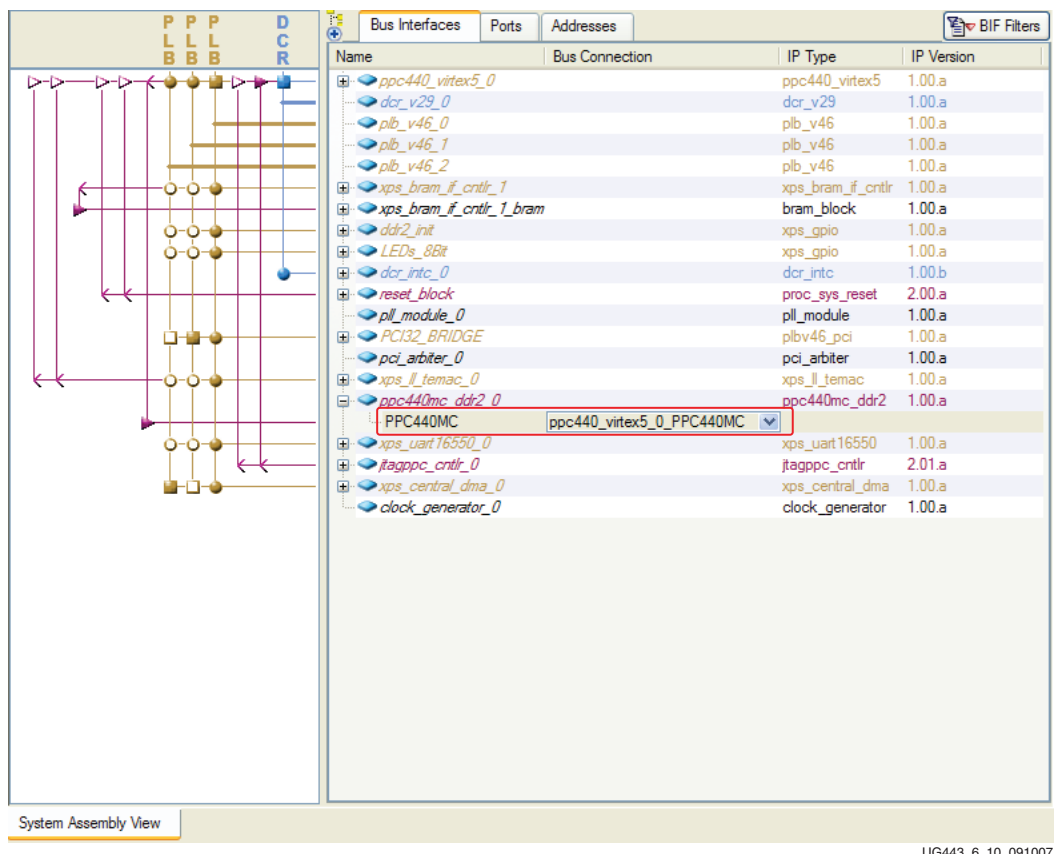


Figure 2-10: PPC440MC DDR2 Bus Interfaces

PPC440MC DDR2 Ports

The PPC440MC DDR2 ports are connected similar to the PLB DDR2 ports. The external ports for DQ and DQS external ports the Name and Net name must be the same. Discussion of connecting the system clocking needed for the memory controller is discussed later in this chapter.

Migrating Ethernet Solution

Adding/Removing Ethernet

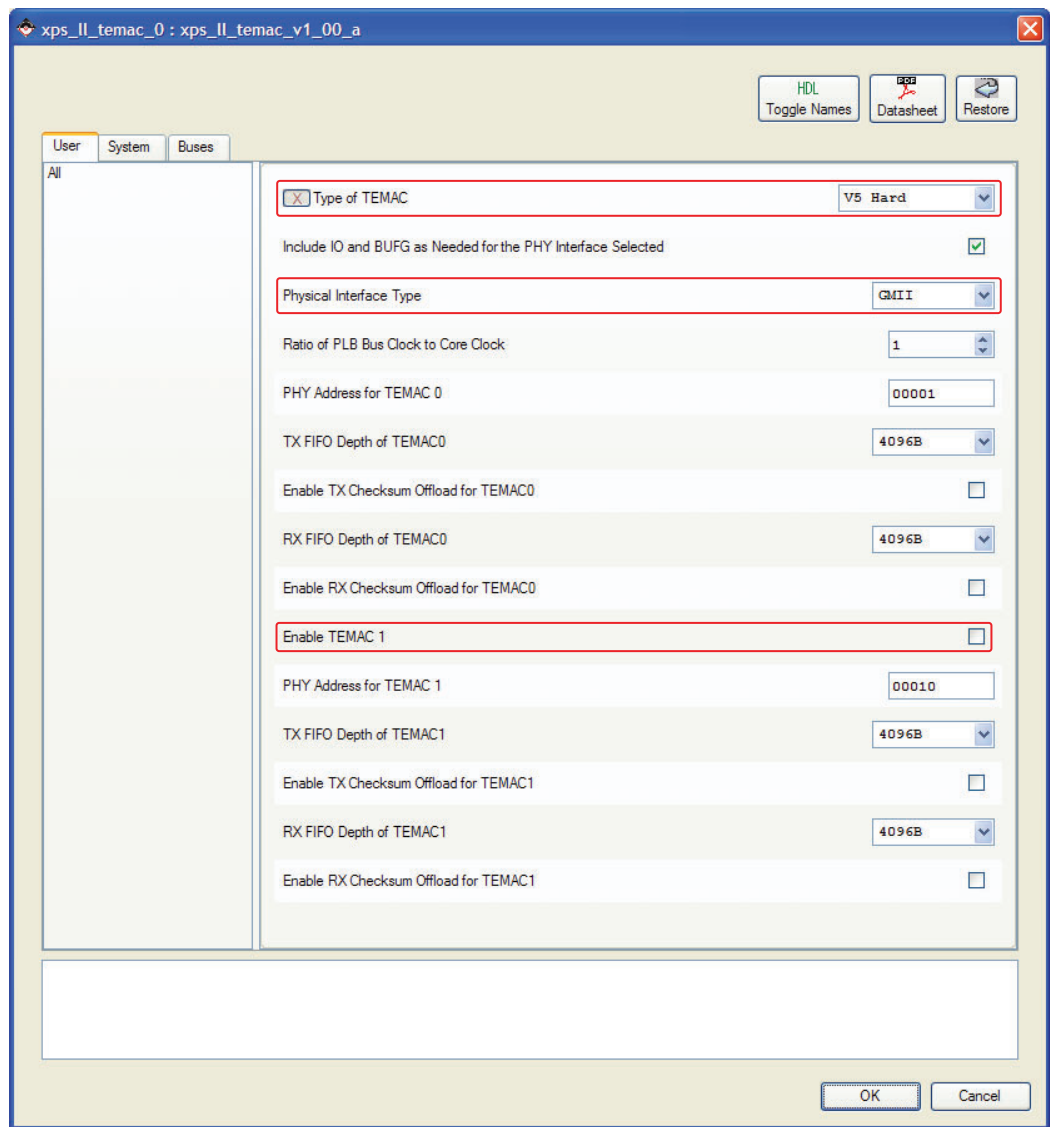
Remove the `TriMode_MAC_GMII` instance by right clicking on **TriMode_MAC_GMII** inside the System Assembly View/Bus Interfaces and clicking on **Delete Instance...**, then click on **Delete instance and its internal ports**.

Add XPS LL TEMAC to the system by expanding the Communication High-Speed tree node inside the IP Catalog tab. Right click on **XPS LocalLink Tri-mode Ethernet Mac** and click on **Add IP**. This creates the `xps_ll_temac_0` instance.

Setting Parameters

Right click on **xps_ll_temac_0** inside the System Assembly View, then select **Configure IP...**

The XPS LL TEMAC is configured to use a single Hard TEMAC by unselecting **Enable TEMAC1**. The GMII interface is used by setting Physical Interface Type to **GMII**. Because the system is for the Virtex-5 FPGA, the Type of TEMAC is set to **V5 Hard**. The above selections are shown in [Figure 2-11](#).



UG443_6_11_091007

Figure 2-11: XPS LL TEMAC Parameters

Setting Bus Interfaces

The slave interface SPLB of the XPS LL TEMAC core is connected to the plb_v46_0 inside the system as shown in [Figure 2-12](#).

The LocalLink bus interface from the XPS LL TEMAC, xps_ll_temac_0_LLINK0, is connected to the LLDMA0 bus interface on the ppc440_virtex5_0, as shown in [Figure 2-12](#).

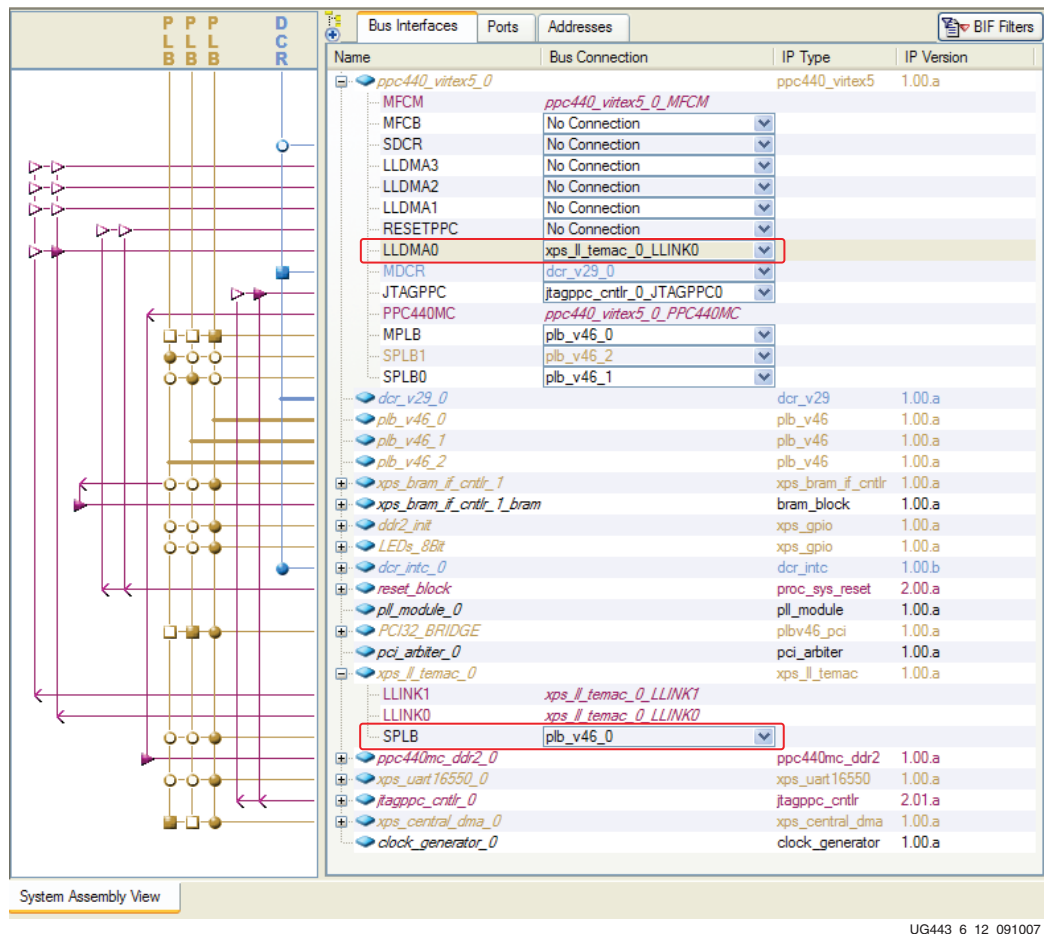


Figure 2-12: XPS LL TEMAC Bus and LocalLink Interfaces

XPS LL TEMAC Ports

The XPS LL TEMAC GMII ports are connected similar to PLB TEMAC GMII ports. The connection of the system clocking for the XPS LL TEMAC is discussed later in this chapter.

Migrating Interrupt Controller to DCR INTC

Add XPS LL TEMAC to the system by expanding the Clock, Reset and Interrupt tree node inside the IP Catalog tab. Right click on **DCR Interrupt Controller** and click on **Add IP**. This creates the `dcr_intc_0` instance.

Setting Bus Interfaces

The SDCR interface from the DCR INTC is connected to the `dcr_v29_0`.

Setting Ports

Expand the tree node for `ppc440_virtex5_0` and `dcr_intc_0`. In the Irq port on the `dcr_intc_0`, click on **Make New Connection** which creates the `dcr_intc_0_Irq`. In the EICC440EXTIRQ input signal from the `ppc440_virtex5_0` is connected to `dcr_intc_0_Irq`.

Other interrupts inside the system should be connect to the Intr port inside the `dcr_intc_0`.

Migrating XPS Peripherals

In the migrated system, the PLB BRAM, OPB INTC, OPB GPIO, OPB PCI and OPB UART Lite cores are replaced with the equivalent XPS BRAM, XPS INTC, XPS GPIO, PLBV46 PCI and XPS UART Lite, respectively.

The addition of these cores is not discussed.

Modifying Clocking/Reset Inside The System

The processor system reset core is replaced with `v2_00_a` version of the core. In addition, the DCMs instantiations in the system are replaced with the PLL Module and the clock generator core.

Processor System Reset

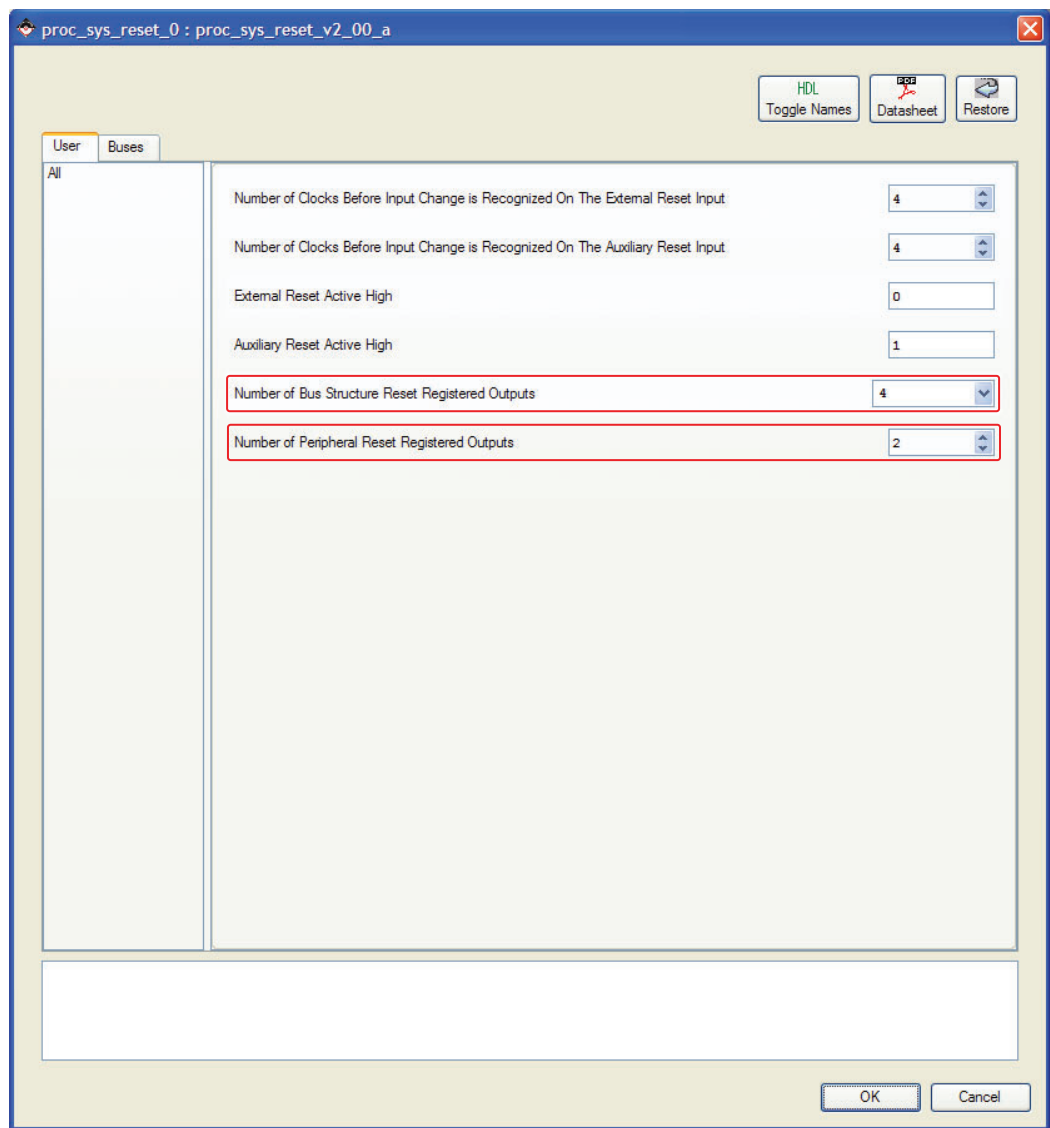
Adding Processor System Reset to Migrated System

In the System Assembly View/Bus Interfaces, delete the `reset_block` instance (Delete instance and its internal ports).

Add the latest version of the processor system reset by expanding the Clock Reset and Interrupt tree node inside the IP Catalog tab. Right click on **Processor System Reset Module** and click on **Add IP**. This creates the `proc_sys_reset_0` instance.

In the System Assembly View, right click on `proc_sys_reset_0`, then select **Configure IP**

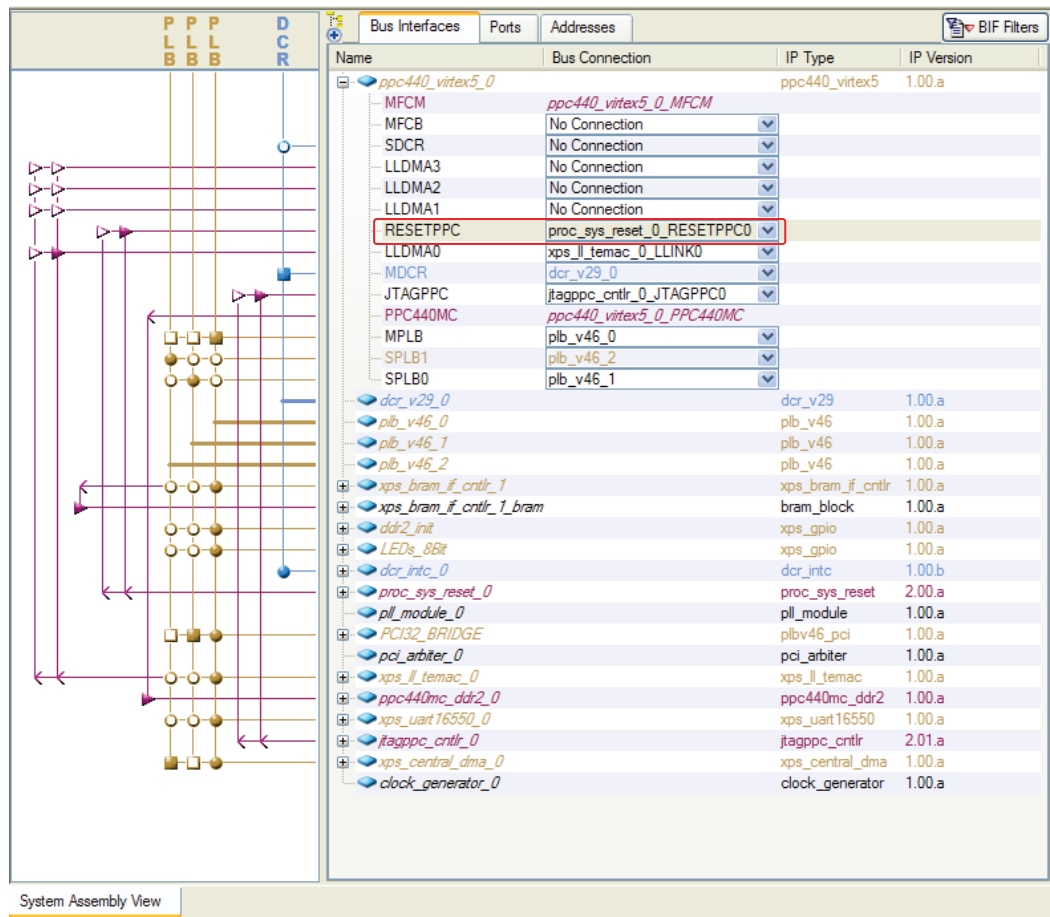
Set External Active High to 0, Number of Bus Structure Reset Registered Outputs to 4, and Number of Peripheral Reset Registered Outputs to 2. The above parameters are shown in Figure 2-13.



UG443_6_13_091007

Figure 2-13: Processor System Reset Parameters

In the Bus Interfaces Tab, expand the ppc440_virtex5_0 tree node. Connect RESETPPC to proc_sys_reset_0_RESETPPC0 as shown in Figure 2-14.



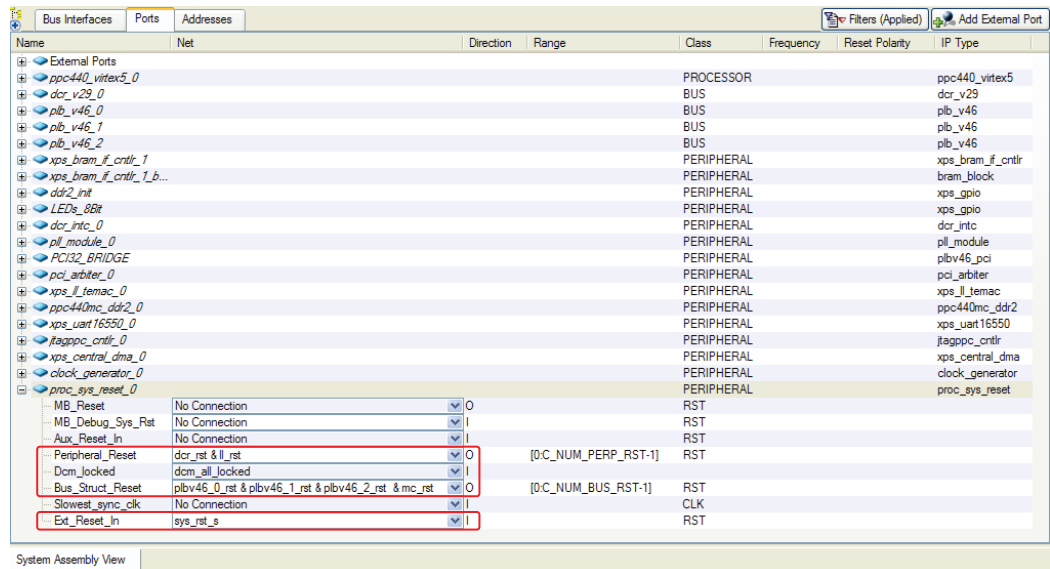
UG443_6_14_091007

Figure 2-14: PowerPC 440 Reset Bus Interface

In the Ports Tab, expand the tree node for `proc_sys_reset_0`.

Set `Peripheral_Reset` to **dcr_rst**, **ll_rst**, `Ext_Reset_in` to **sys_rst_s**, `Bus_Struct_Reset` to **plbv46_0_rst**, **plbv46_1_rst**, **plbv46_2_rst** & **mc_rst**. In addition, set `Dcm_locked` to **dcm_all_locked**. The `dcm_all_locked` input signal is defined in the clocking section.

The above port connections are not shown in Figure 2-15.



Name	Net	Direction	Range	Class	Frequency	Reset Polarity	IP Type
External Ports							
ppc440_virtex5_0				PROCESSOR			ppc440_virtex5
dcr_v29_0				BUS			dcr_v29
plb_v46_0				BUS			plb_v46
plb_v46_1				BUS			plb_v46
plb_v46_2				BUS			plb_v46
xps_bram_if_ctrlr_1				PERIPHERAL			xps_bram_if_ctrlr
xps_bram_if_ctrlr_1_b...				PERIPHERAL			bram_block
ddr2_int				PERIPHERAL			xps_gpio
LEDs_8Bit				PERIPHERAL			xps_gpio
dcr_intc_0				PERIPHERAL			dcr_intc
pll_module_0				PERIPHERAL			pll_module
PC132_BRIDGE				PERIPHERAL			plbv46_pci
pcl_arbiter_0				PERIPHERAL			pcl_arbiter
xps_ll_temac_0				PERIPHERAL			xps_ll_temac
ppc440mc_ddr2_0				PERIPHERAL			ppc440mc_ddr2
xps_uart16550_0				PERIPHERAL			xps_uart16550
itagppc_ctrlr_0				PERIPHERAL			itagppc_ctrlr
xps_central_dma_0				PERIPHERAL			xps_central_dma
clock_generator_0				PERIPHERAL			clock_generator
proc_sys_reset_0				PERIPHERAL			proc_sys_reset
MB_Reset	No Connection		0	RST			
MB_Debug_Sys_Rst	No Connection		1	RST			
Aux_Reset_in	No Connection		1	RST			
Peripheral_Reset	dcr_rst & ll_rst		0	RST	[0:C_NUM_PERP_RST-1]		
Dcm_locked	dcm_all_locked		1				
Bus_Struct_Reset	plbv46_0_rst & plbv46_1_rst & plbv46_2_rst & mc_rst		0	RST	[0:C_NUM_BUS_RST-1]		
Slowest_sync_clk	No Connection			CLK			
Ext_Reset_in	sys_rst_s		1	RST			

Figure 2-15: Processor System Reset Connections

Expand the tree nodes for `plb_v46_0`, `plb_v46_1`, and `plb_v46_2` in the System Assembly View/Port. Connect the `SYS_Rst` in the bus instances to **plbv46_0_rst**, **plbv46_1_rst**, and **plbv46_2_rst**, respectively.

For master and slave peripherals connected to the PLB v4.6, the peripherals get vectorized resets for each master and slave connection from the bus core.

Expand the tree node for `ppc440_mc_ddr2_0` and set the `mc_mreset` port to **mc_rst**.

Expand the tree node for `xps_ll_temac_0` and set the `LlinkTemac0_RST` port to **ll_rst**.

Expand the tree node for `dcr_intc_0` and set the `DCR_Rst` to **dcr_rst**.

Clock Generator and PLL Module

Overview of Existing Clocking Scheme

At the outset, the system contains 2 cascading DCMs. The first DCM provides the 100 MHz clock for the bus. The 200 MHz clock provides the processor clock, PLB DDR2 Device_Clk0, and the RCLK to the OPB PCI. In addition, the first DCM provides the 125 MHz clock for the Hard TEMAC, the 25 MHz clock for the Cal_Clk for PLB DDR2, and the 33 MHz clock for the PCLK to the OPB PCI.

The second DCM provides the 200 MHz Device_Clk90_in_n for the PLB DDR2 memory controller.

With the PLB DDR2 clocks, inverters are needed for the Device_Clk0 and Device_Clk90_in_n from the DCMs to provide the PLB DDR2 Device_Clk0_n and Device_Clk90_in_n clocks.

Clocking Requirements of Migrated System

The migrated system uses the PPC440 processor block with a processor frequency of 400 MHz and the PPC440MC with a frequency of 200 MHz. The cross bar is set to 200 MHz. In addition, the MPLB, SPLB0, SPLB1 and DCR ports are set to 100 MHz.

In addition, the PPC440MC DDR2 and PLBV46 PCI needs a 200 MHz clock for the IDELAY Controllers, while the PPC440MC DDR2 needs a 200 MHz CLK0 and CLK90 clocks. Also, a 125 MHz clock is needed for the Hard TEMAC, and a 33 MHz clock is needed for PCI.

The clocks for the PLBV46 PCI core are not discussed.

The above system clocking is shown in [Figure 2-16](#).

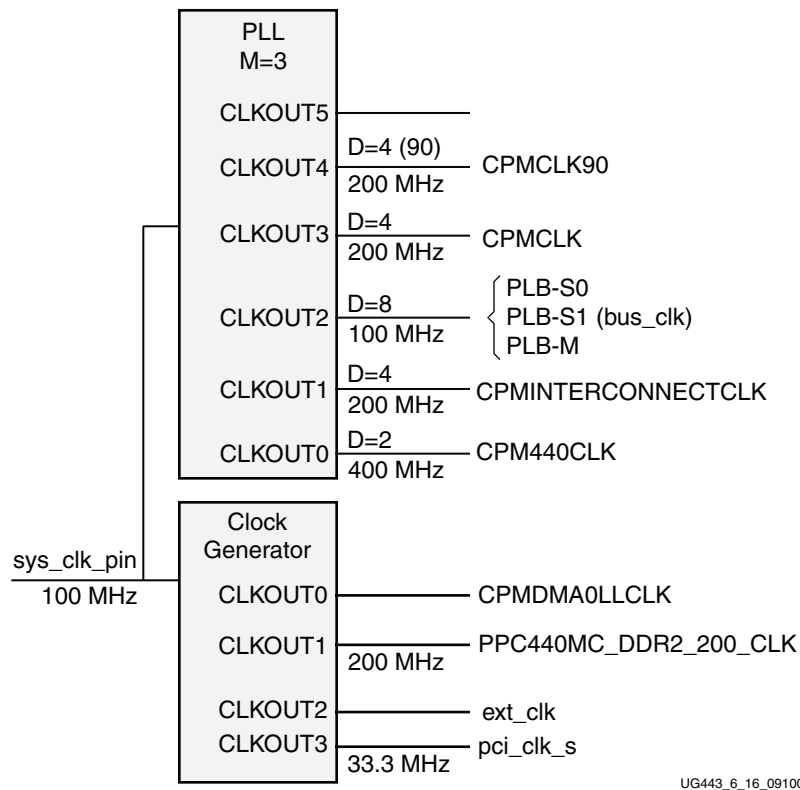


Figure 2-16: System Clocking Block Diagram

In the System Assembly View/Bus Interfaces, delete the `dcm_0` and `dcm_1` instances (Delete instance and its internal ports).

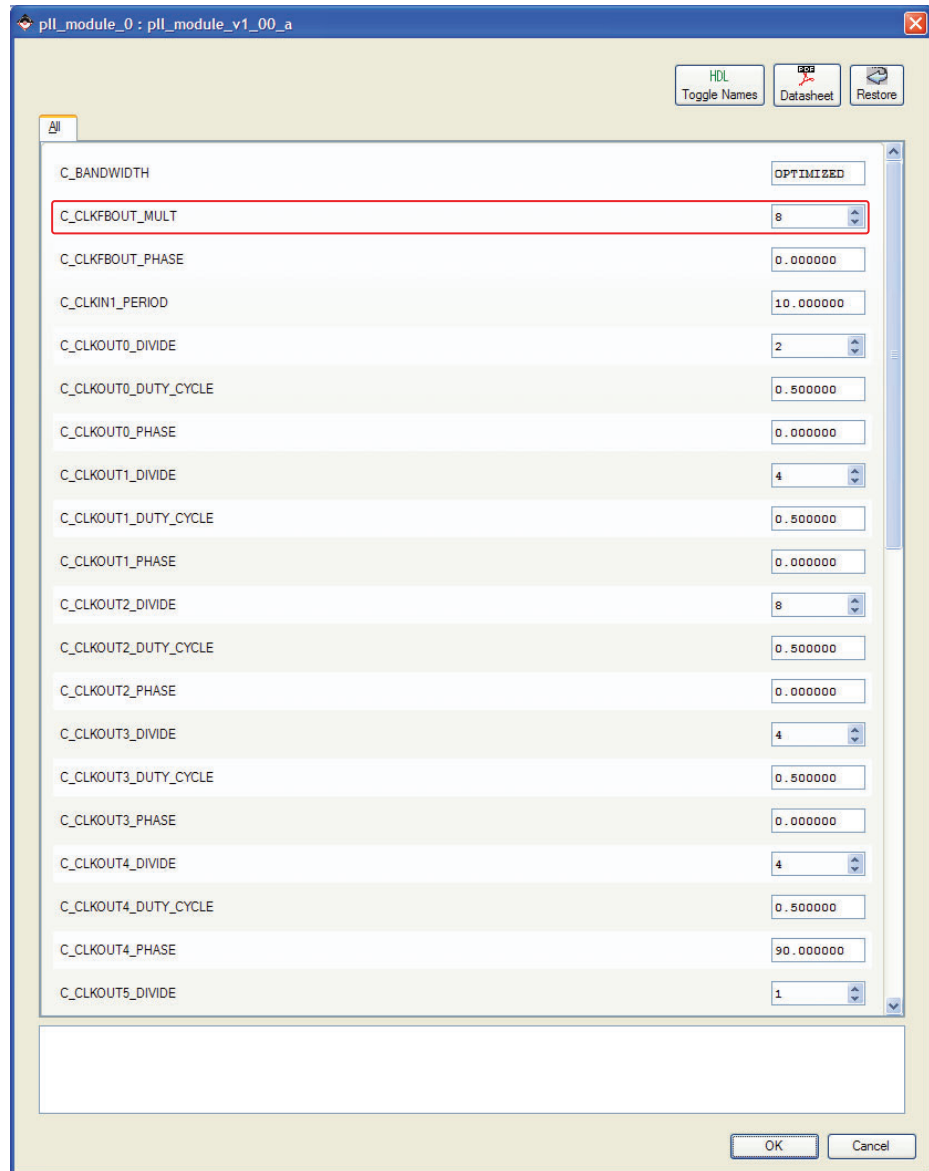
Configuring PLL Module Inside The Migrated System

Add the PLL Module by expanding the Clock Reset and Interrupt or the Global Peripheral Repository tree node inside the IP Catalog tab. Right click on **PLL for PPC440 Clocks** and click on **Add IP**. This creates the `pll_module_0` instance inside the system.

Right click on `pll_module_0` in the System Assembly View and select **Configure IP ...**.

The PLL Module is set for clocking block diagram shown in [Figure 2-16](#).

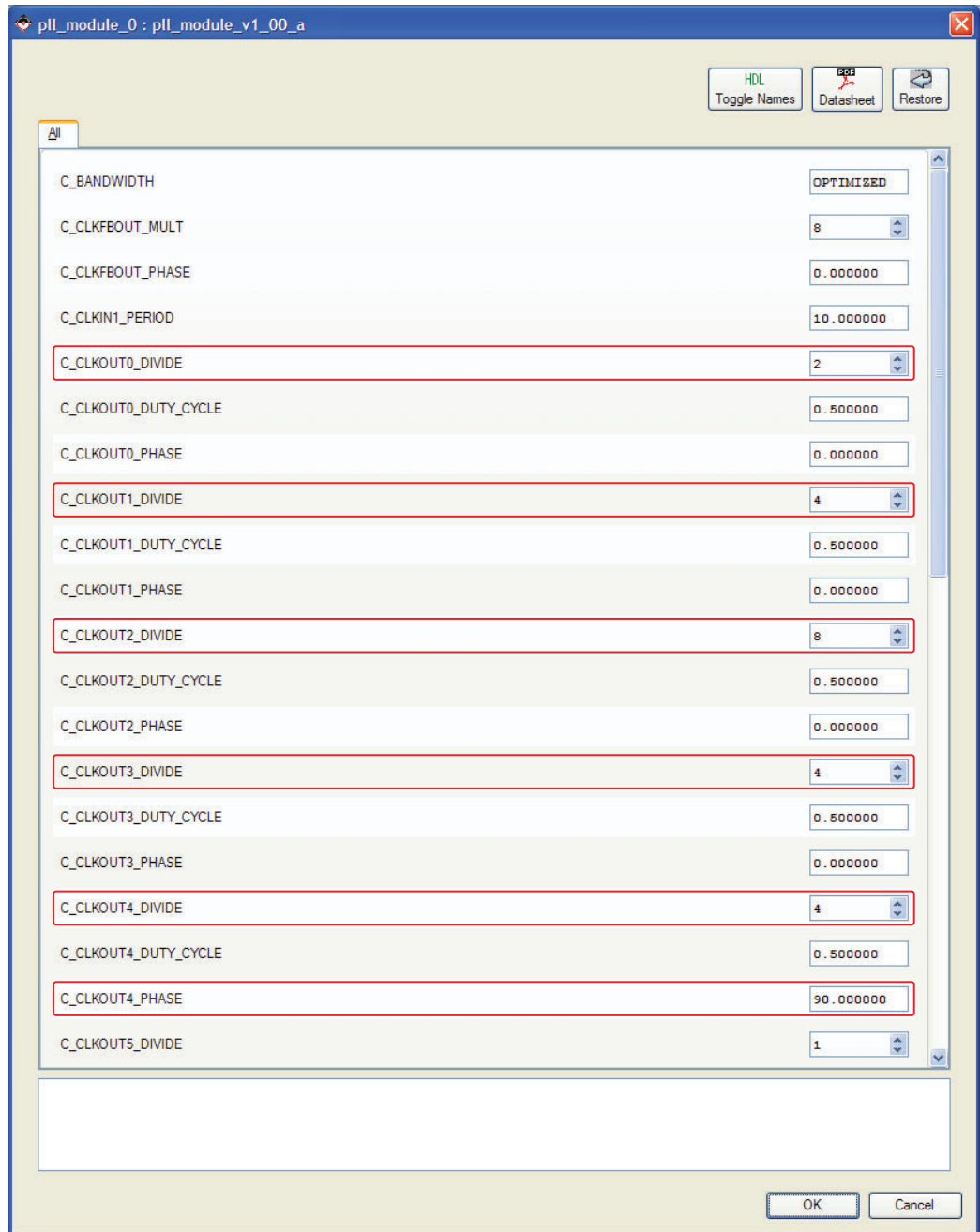
Set C_CLKFBOUT_MULT to **8**. This provides the module to produce 400, 200, and 100 MHz outputs by divided parameters on the individual ports. The above settings are shown in [Figure 2-17](#)



UG443_6_17_091007

Figure 2-17: PLL Module Multiply Parameter

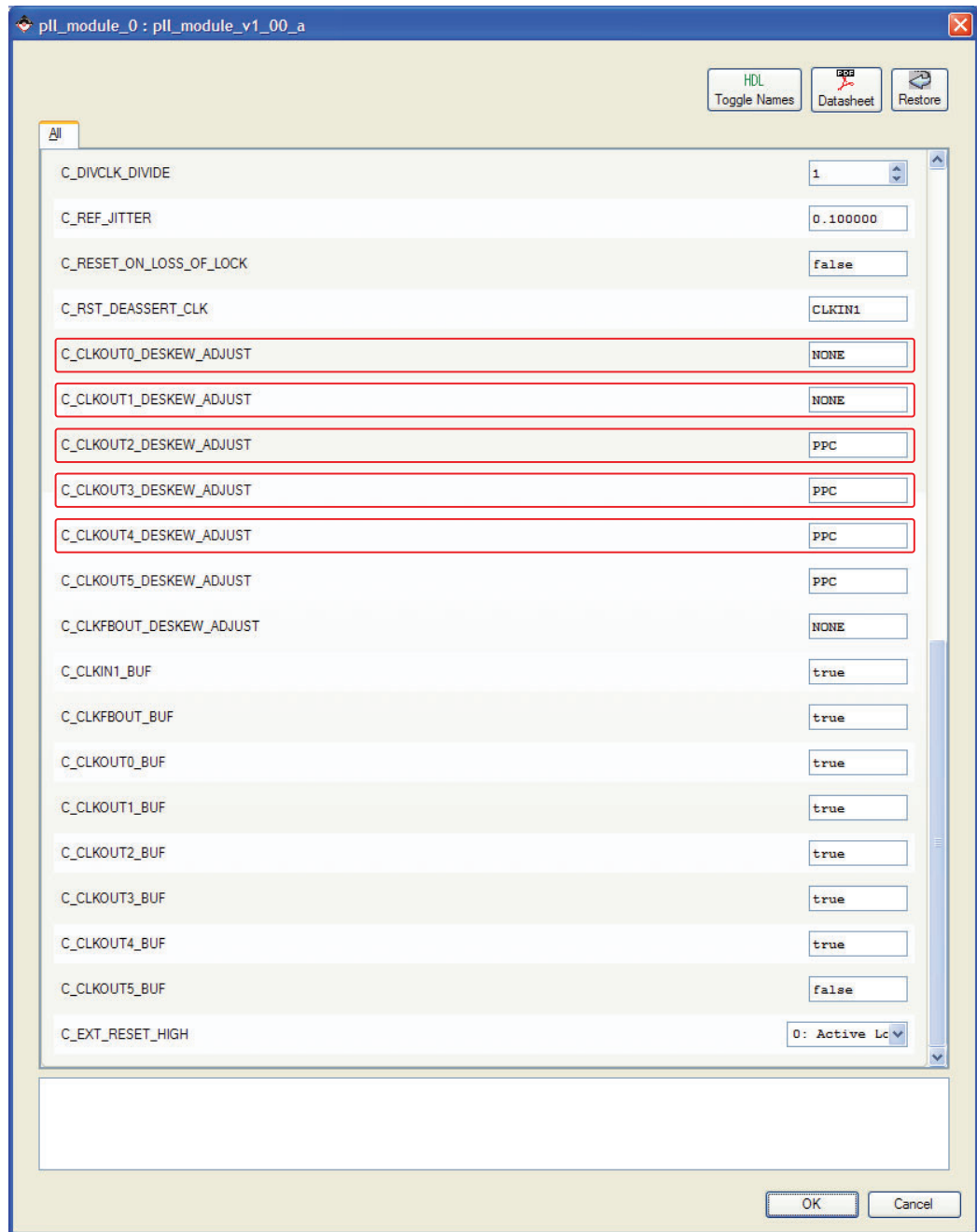
Set C_CLKOUT0_DIVIDE to **2**, C_CLKOUT1_DIVIDE to **4**, C_CLKOUT2_DIVIDE to **8**, C_CLKOUT3_DIVIDE to **4**, and C_CLKOUT4_DIVIDE to **4**. In addition, C_CLKOUT4_PHASE is set to **90.000000** because this output is for phase shift of the PPC440MC DDR2. The above settings are shown in [Figure 2-18](#).



UG443_6_18_091007

Figure 2-18: PLL Module Divide/Phase Parameters

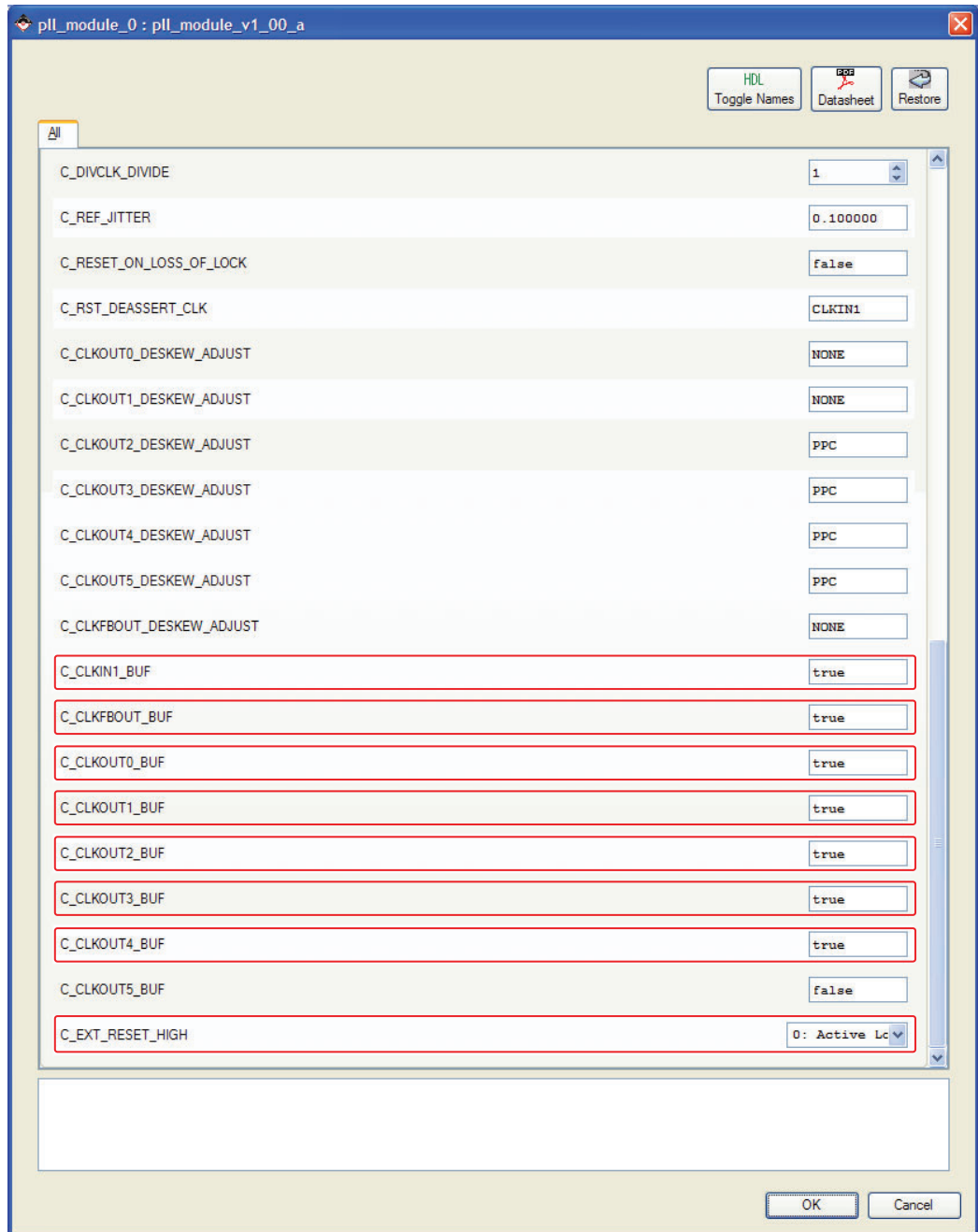
Set C_CLKOUT0_DESKEW_ADJUST to **NONE**, C_CLKOUT1_DESKEW_ADJUST to **NONE**, C_CLKOUT2_DESKEW_ADJUST to **PPC**, C_CLKOUT3_DESKEW_ADJUST to **PPC**, and C_CLKOUT4_DESKEW_ADJUST to **PPC**. The above settings are shown in Figure 2-19.



UG443_6_19_091007

Figure 2-19: PLL Module Deskew Parameters

Set C_CLKIN1_BUF to **true**, C_CLKFBOUT_BUF to **true**, C_CLKOUT0_BUF to **true**, C_CLKOUT1_BUF to **true**, C_CLKOUT2_BUF to **true**, C_CLKOUT3_BUF to **true**, and C_CLKIN4_BUF to **true**. In addition, set C_EXT_RESET_HIGH to **0: Active Low**. The above settings are shown in [Figure 2-20](#).



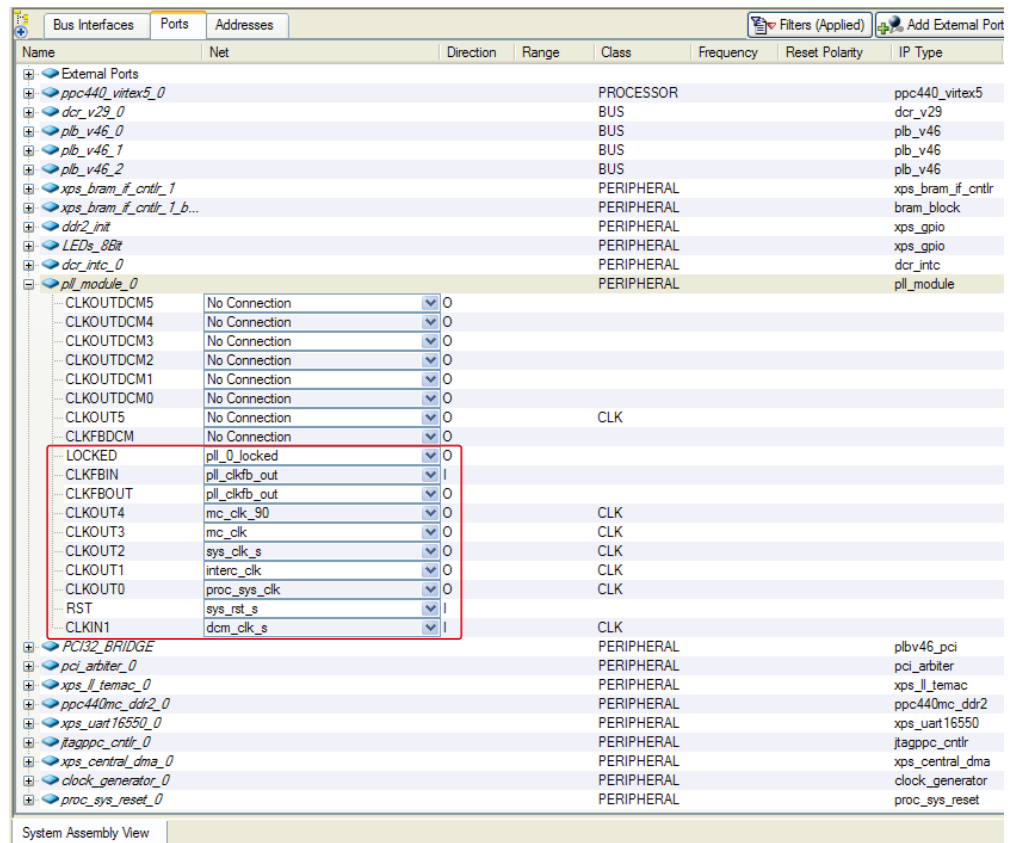
UG443_6_20_091007

Figure 2-20: PLL Module Buffer/Reset Parameters

Expand the pll_module_0 tree node in the System Assembly View/Port tab. Make the connections as shown in Table 2-1 and in Figure 2-21.

Table 2-1: PLL Module Ports and Connections

Port Name	Connection	I/O	Description
CLKFBIN	pll_clkfb_out	I	Feedback clock input.
CLKFBOUT	pll_clkfb_out	O	Feedback clock output.
CLKOUT0	proc_sys_clk	O	PowerPC 440 processor clock output.
CLKOUT1	interc_clk	O	Interconnect clock output.
CLKOUT2	sys_clk_s	O	Clock output for MPLB, SPLB0, SPLB1 and DCR.
CLKOUT3	mc_clk	O	MC clock output.
CLKOUT4	mc_clk_90	O	MC clock 90 output.
RST	sys_rst_s	I	Asynchronous reset signal.
LOCKED	pll_0_locked	O	An high output when PLL achieves phase alignment.
CLKIN1	dcm_clk_s	I	Primary clock input.

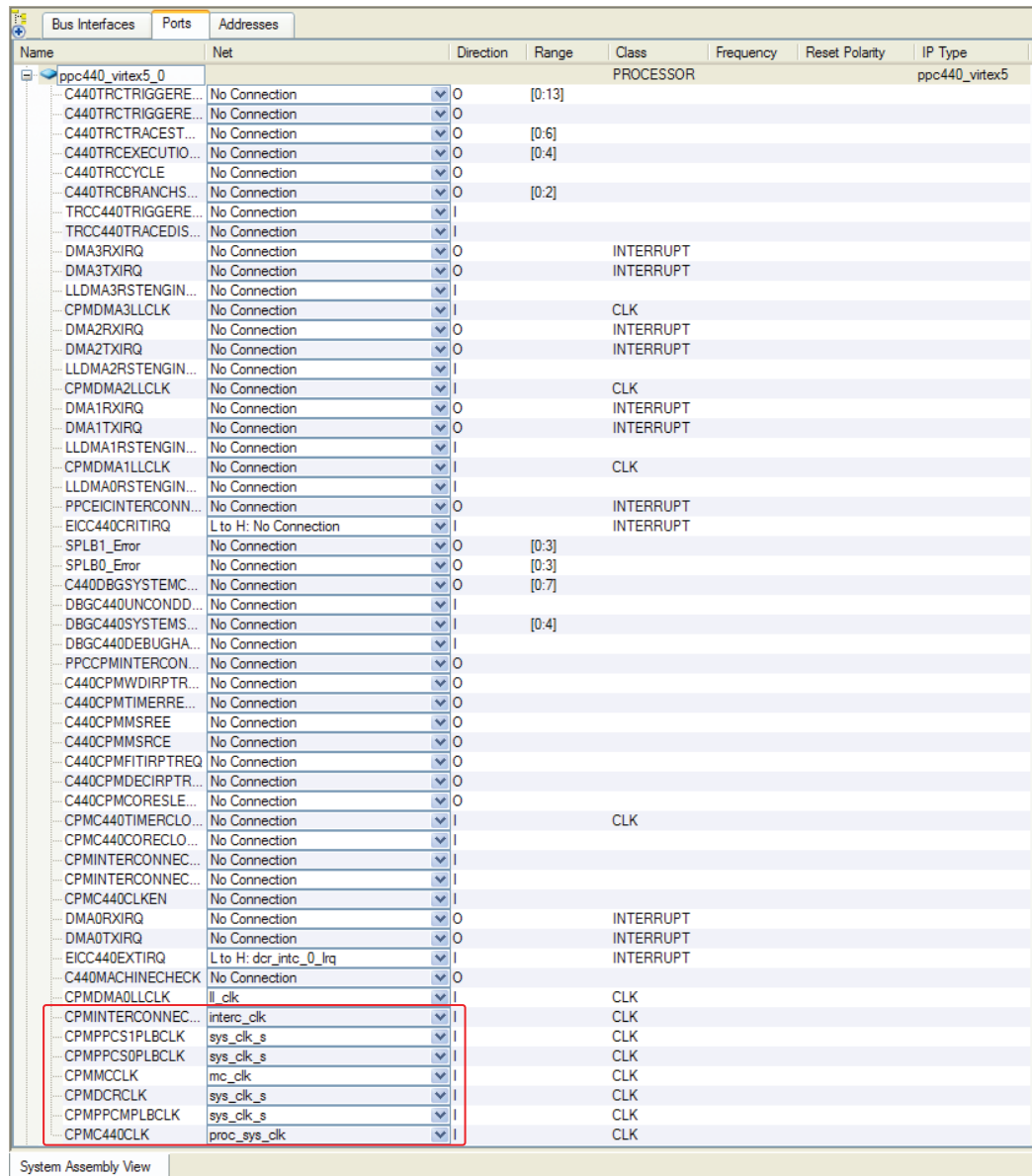


UG443_6_21_091007

Figure 2-21: PLL Module Port Connections

Expand the ppc440_0 tree node in the Port tab. Connect CPMPPCS1PLBCLK, CPMPPCS0PLBCLK, CPMDCRCLK, CPMPPCM to sys_clk_s. Connect the CPMINTERCONNECTCLK to interc_clk, CPMCCCLK to mc_clk and CPMC440CLK to proc_sys_clk.

The above connections are shown [Figure 2-22](#).



Name	Net	Direction	Range	Class	Frequency	Reset Polarity	IP Type
ppc440_virtex5_0				PROCESSOR			ppc440_virtex5
C440TRCTRIGGERE...	No Connection	O	[0:13]				
C440TRCTRIGGERE...	No Connection	O					
C440TRCTRACEST...	No Connection	O	[0:6]				
C440TRCEEXECUTIO...	No Connection	O	[0:4]				
C440TRCCYCLE	No Connection	O					
C440TRCBRANCHS...	No Connection	O	[0:2]				
TRCC440TRIGGERE...	No Connection	I					
TRCC440TRACEDIS...	No Connection	I					
DMA3RXIRQ	No Connection	O		INTERRUPT			
DMA3TXIRQ	No Connection	O		INTERRUPT			
LLDMA3RSTENGIN...	No Connection	I					
CPMDMA3LLCLK	No Connection	I		CLK			
DMA2RXIRQ	No Connection	O		INTERRUPT			
DMA2TXIRQ	No Connection	O		INTERRUPT			
LLDMA2RSTENGIN...	No Connection	I					
CPMDMA2LLCLK	No Connection	I		CLK			
DMA1RXIRQ	No Connection	O		INTERRUPT			
DMA1TXIRQ	No Connection	O		INTERRUPT			
LLDMA1RSTENGIN...	No Connection	I					
CPMDMA1LLCLK	No Connection	I		CLK			
LLDMA0RSTENGIN...	No Connection	I					
PPCEICINTERCONN...	No Connection	O		INTERRUPT			
EICC440CRITIRQ	L to H: No Connection	I		INTERRUPT			
SPLB1_Error	No Connection	O	[0:3]				
SPLB0_Error	No Connection	O	[0:3]				
C440DBGSYSTEMC...	No Connection	O	[0:7]				
DBG440UNCOND...	No Connection	I					
DBG440SYSTEMS...	No Connection	I	[0:4]				
DBG440DEBUGHA...	No Connection	I					
PPCCPMINTERCON...	No Connection	O					
C440CPMWDIRPTR...	No Connection	O					
C440CPMTIMERRE...	No Connection	O					
C440CPMMSREE	No Connection	O					
C440CPMMSRCE	No Connection	O					
C440CPMFITIRPTREQ	No Connection	O					
C440CPMDECIRPTR...	No Connection	O					
C440CPMCORESLE...	No Connection	O					
CPMC440TIMERCLC...	No Connection	I		CLK			
CPMC440CORECLO...	No Connection	I					
CPMINTERCONN...	No Connection	I					
CPMINTERCONN...	No Connection	I					
CPMC440CLKEN	No Connection	I					
DMA0RXIRQ	No Connection	O		INTERRUPT			
DMA0TXIRQ	No Connection	O		INTERRUPT			
EICC440EXTIRQ	L to H: dcr_intc_0_irq	I		INTERRUPT			
C440MACHINECHECK	No Connection	O					
CPMDMA0LLCLK	ll_clk	I		CLK			
CPMINTERCONN...	interc_clk	I		CLK			
CPMPPCS1PLBCLK	sys_clk_s	I		CLK			
CPMPPCS0PLBCLK	sys_clk_s	I		CLK			
CPMMCCLK	mc_clk	I		CLK			
CPMDCRCLK	sys_clk_s	I		CLK			
CPMPPCMPLBCLK	sys_clk_s	I		CLK			
CPMC440CLK	proc_sys_clk	I		CLK			

UG443_6_22_091007

Figure 2-22: PPC440 Clock Port Connections

Expand the tree node for ppc440_mc_ddr2_0 in the Port tab. Connect mc_mibclk to mc_clk and mi_mccclk90 to mc_clk_90.

The above connections are shown [Figure 2-23](#).

Name	Net	Direction	Range	Class	Frequency	Reset Polarity	IP Type
External Ports							
ppc440_virtex5_0				PROCESSOR			ppc440_virtex5
dcr_v29_0				BUS			dcr_v29
plb_v46_0				BUS			plb_v46
plb_v46_1				BUS			plb_v46
plb_v46_2				BUS			plb_v46
xps_bram_if_ctrlr_1				PERIPHERAL			xps_bram_if_ctrlr
xps_bram_if_ctrlr_1_b...				PERIPHERAL			bram_block
ddr2_int				PERIPHERAL			xps_gpio
LEDs_8Bit				PERIPHERAL			xps_gpio
dcr_intc_0				PERIPHERAL			dcr_intc
pll_module_0				PERIPHERAL			pll_module
PCI32_BRIDGE				PERIPHERAL			plb46_pci
pci_arbiter_0				PERIPHERAL			pci_arbiter
xps_ll_temac_0				PERIPHERAL			xps_ll_temac
ppc440mc_ddr2_0				PERIPHERAL			ppc440mc_ddr2
mc_miwritereadytoac...	No Connection		0				
mc_miwllbebusy	No Connection		0				
mc_mireaddataparity	No Connection		0		[(C_DDR_DM_WIDTH*2)-1:0]		
mc_mibusy	No Connection		0				
mi_mcwttedataparity	No Connection		1		[(C_DDR_DM_WIDTH*2)-1:0]		
DDR2_CK_N	fpga_0_DDR2_SDRAM_32Mx64_DDR_ClkN		0		[(C_NUM_CLK_PAIRS-1):0]		
DDR2_CK	fpga_0_DDR2_SDRAM_32Mx64_DDR_Clk		0		[(C_NUM_CLK_PAIRS-1):0]		
DDR2_DM	fpga_0_DDR2_SDRAM_32Mx64_DDR_DM		0		[(C_DDR_DM_WIDTH-1):0]		
DDR2_CKE	fpga_0_DDR2_SDRAM_32Mx64_DDR_CKE		0		[(C_NUM_RANKS_MEM-1):0]		
DDR2_ODT	fpga_0_DDR2_SDRAM_32Mx64_DDR_ODT		0		[(C_DDR2_ODT_WIDTH-1):0]		
DDR2_CS_N	fpga_0_DDR2_SDRAM_32Mx64_DDR_CSn		0		[(C_NUM_RANKS_MEM-1):0]		
DDR2_WE_N	fpga_0_DDR2_SDRAM_32Mx64_DDR_WEn		0				
DDR2_CAS_N	fpga_0_DDR2_SDRAM_32Mx64_DDR_CASn		0				
DDR2_RAS_N	fpga_0_DDR2_SDRAM_32Mx64_DDR_RASn		0				
DDR2_BA	fpga_0_DDR2_SDRAM_32Mx64_DDR_BankAddr		0		[(C_DDR_BAWIDTH-1):0]		
DDR2_A	fpga_0_DDR2_SDRAM_32Mx64_DDR_Addr		0		[(C_DDR_RAWIDTH-1):0]		
DDR2_DQS_N	DDR2_DQS_N		IO		[(C_DDR_DQS_WIDTH-1):0]		
DDR2_DQS	DDR2_DQS		IO		[(C_DDR_DQS_WIDTH-1):0]		
DDR2_DQ	DDR2_DQ		IO		[(C_DDR_DWIDTH-1):0]		
mi_mccclk_200	No Connection		1				
mi_mcreaset	mc_rst		1			RST	
mi_mccclk90	mc_clk_90		1				
mc_mibclk	mc_clk		1				
xps_uart16550_0				PERIPHERAL			xps_uart16550
itagppc_ctrlr_0				PERIPHERAL			itagppc_ctrlr
xps_central_dma_0				PERIPHERAL			xps_central_dma
clock_generator_0				PERIPHERAL			clock_generator
proc_sys_reset_0				PERIPHERAL			proc_sys_reset

UG443_6_23_091007

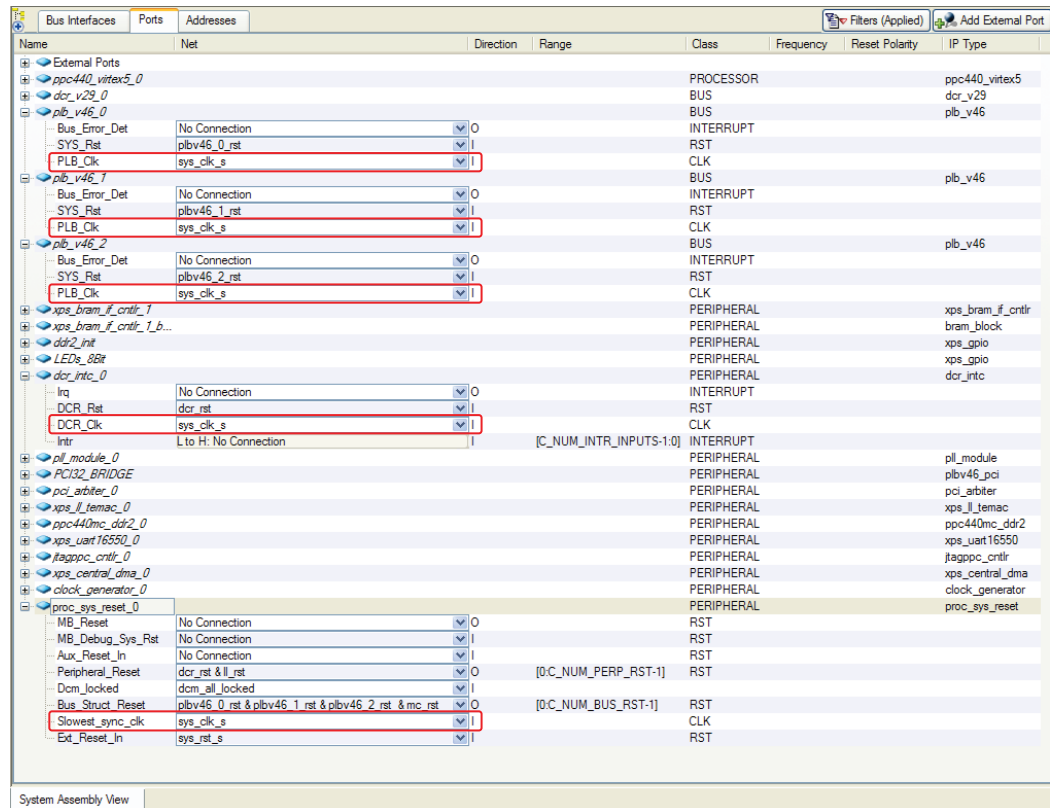
Figure 2-23: PPC440MC DDR2 Clock Port Connections

Expand the tree nodes for plb_v46_0, plb_v46_1, and, plb_v46_2 in the System Assembly View/Port tab. Connect the PLB_Clk in all bus instances to sys_clk_s.

Expand the tree node for proc_sys_reset_0. Connect Slowest_sync_clk to sys_clk_s.

Expand the tree node for dcr_intc_0. Connect the DCR_Clk to sys_clk_s.

The above connections are shown in [Figure 2-24](#).



Name	Net	Direction	Range	Class	Frequency	Reset Polarity	IP Type
External Ports							
ppc440_virtex5_0				PROCESSOR			ppc440_virtex5
dcr_v29_0				BUS			dcr_v29
plb_v46_0				BUS			plb_v46
Bus_Error_Det	No Connection	0		INTERRUPT			
SYS_Rst	plbv46_0_rst	1		RST			
PLB_Clk	sys_clk_s	1		CLK			
plb_v46_1				BUS			plb_v46
Bus_Error_Det	No Connection	0		INTERRUPT			
SYS_Rst	plbv46_1_rst	1		RST			
PLB_Clk	sys_clk_s	1		CLK			
plb_v46_2				BUS			plb_v46
Bus_Error_Det	No Connection	0		INTERRUPT			
SYS_Rst	plbv46_2_rst	1		RST			
PLB_Clk	sys_clk_s	1		CLK			
xps_bram_if_ctrl_1				PERIPHERAL			xps_bram_if_ctrl
xps_bram_if_ctrl_1_b...				PERIPHERAL			bram_block
ddr2_init				PERIPHERAL			xps_gpio
LEDs_8Bit				PERIPHERAL			xps_gpio
dcr_intc_0				PERIPHERAL			dcr_intc
Irq	No Connection	0		INTERRUPT			
DCR_Rst	dcr_rst	1		RST			
DCR_Clk	sys_clk_s	1		CLK			
Intr	L to H: No Connection	1	[C_NUM_INTR_INPUTS-1:0]	INTERRUPT			
pll_module_0				PERIPHERAL			pll_module
PCI32_BRIDGE				PERIPHERAL			plbv46_pci
pci_arbiter_0				PERIPHERAL			pci_arbiter
xps_ll_temac_0				PERIPHERAL			xps_ll_temac
ppc440mc_ddr2_0				PERIPHERAL			ppc440mc_ddr2
xps_uart16550_0				PERIPHERAL			xps_uart16550
jtagppc_ctrl_0				PERIPHERAL			jtagppc_ctrl
xps_central_dma_0				PERIPHERAL			xps_central_dma
clock_generator_0				PERIPHERAL			clock_generator
proc_sys_reset_0				PERIPHERAL			proc_sys_reset
MB_Reset	No Connection	0		RST			
MB_Debug_Sys_Rst	No Connection	1		RST			
Aux_Reset_In	No Connection	1		RST			
Peripheral_Reset	dcr_rst & ll_rst	0	[0:C_NUM_PERP_RST-1]	RST			
Dom_Locked	dcm_all_locked	1		RST			
Bus_Struct_Reset	plbv46_0_rst & plbv46_1_rst & plbv46_2_rst & mc_rst	0	[0:C_NUM_BUS_RST-1]	RST			
Slowest_sync_clk	sys_clk_s	1		CLK			
Ext_Reset_In	sys_rst_s	1		RST			

UG443_6_24_091007

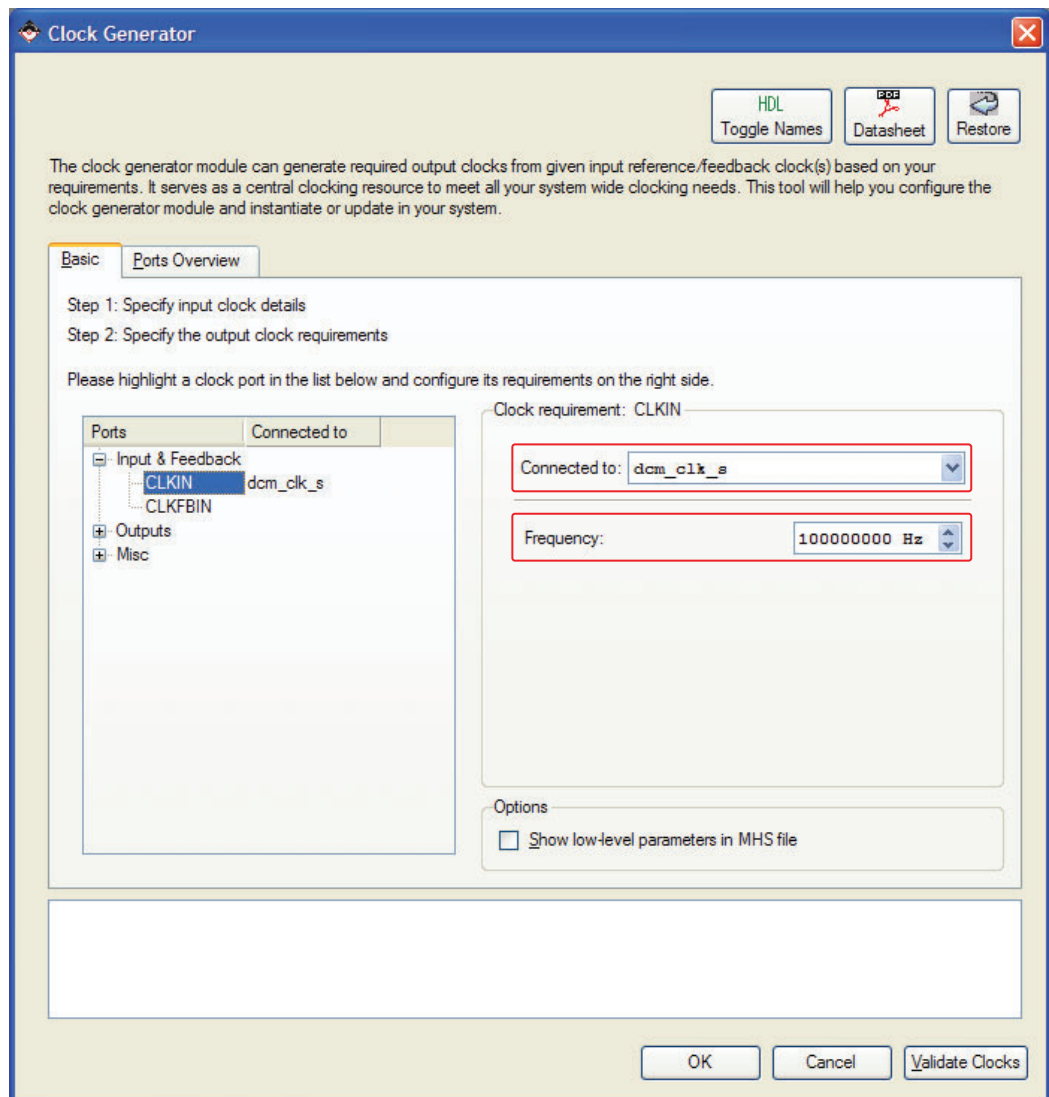
Figure 2-24: PLB v46 Instances Clock Port Connections

Configuring Clock Generator Inside The Migrated System

Add the clock generator by expanding the Clock Reset and Interrupt tree node inside the IP Catalog tab. Right click on **Clock Generator** and click on **Add IP**. This creates the clock_generator_0 instance inside the system.

Right click on **clock_generator_0** in the System Assembly View and select **Configure IP**

The inputs and outputs for the Clock Generator are set up in the Basic tab. Under the Ports, expand the Input & Feedback tree node. Click on **CLKIN**. On the right side of the window, select **Connect to: dcm_clk_s**. In addition, change the Frequency: to **100000000 Hz** as shown in [Figure 2-25](#).



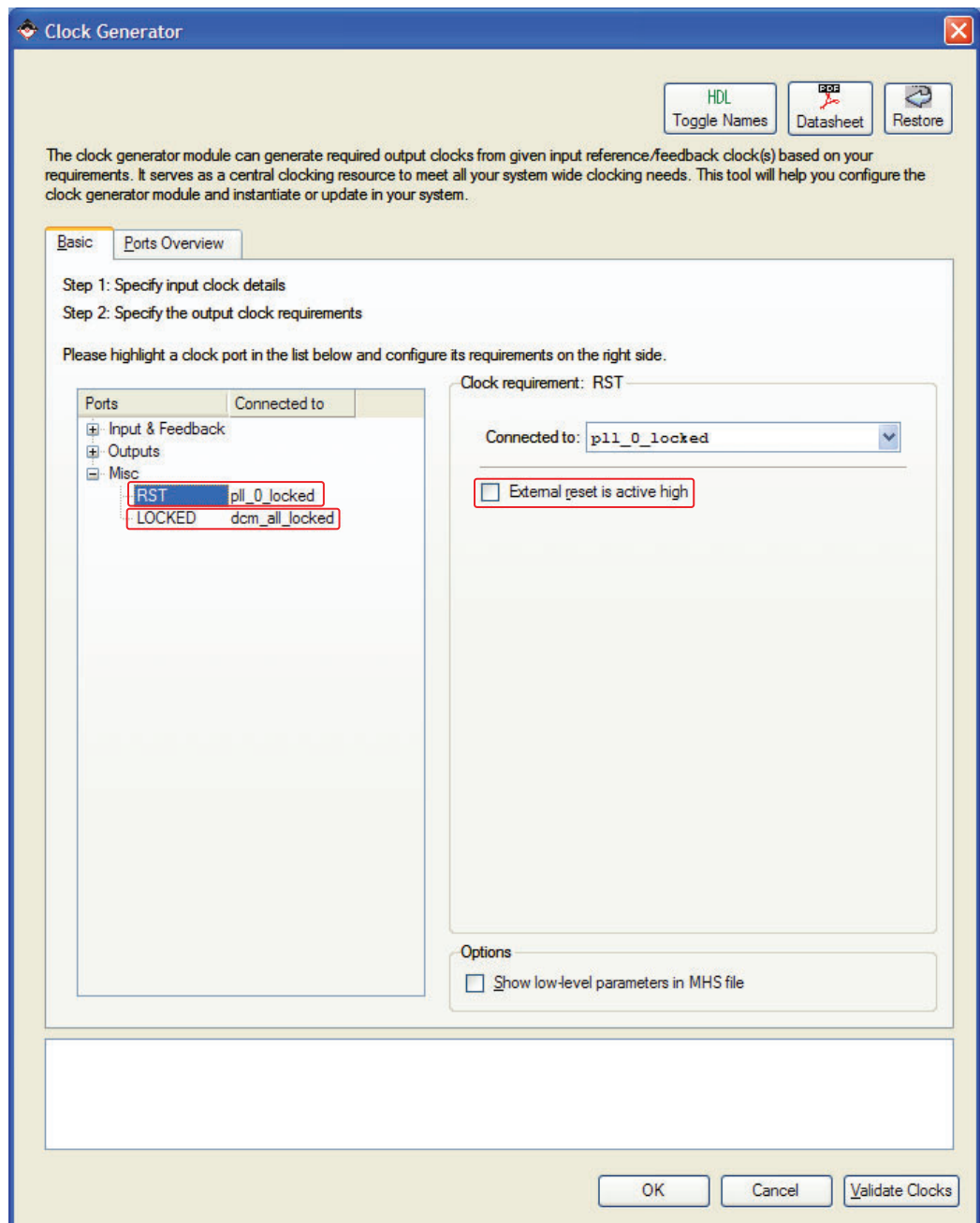
UG443_6_25_091007

Figure 2-25: Clock Generator Input & Feedback

Under **Ports Overview**, expand the **Misc** tree node which is near the bottom. Click on **RST**. On the right side of the window, select **Connect to: pll_0_locked**. Do not select **External reset is active high** because the reset of the clock module is in reset until the PLL module achieves phase alignment.

Click on **LOCKED**. On the right side of the window, select **Connect to: dcm_all_locked**. This output is connected to an input on the processor system reset module.

The above connections are shown in [Figure 2-26](#).



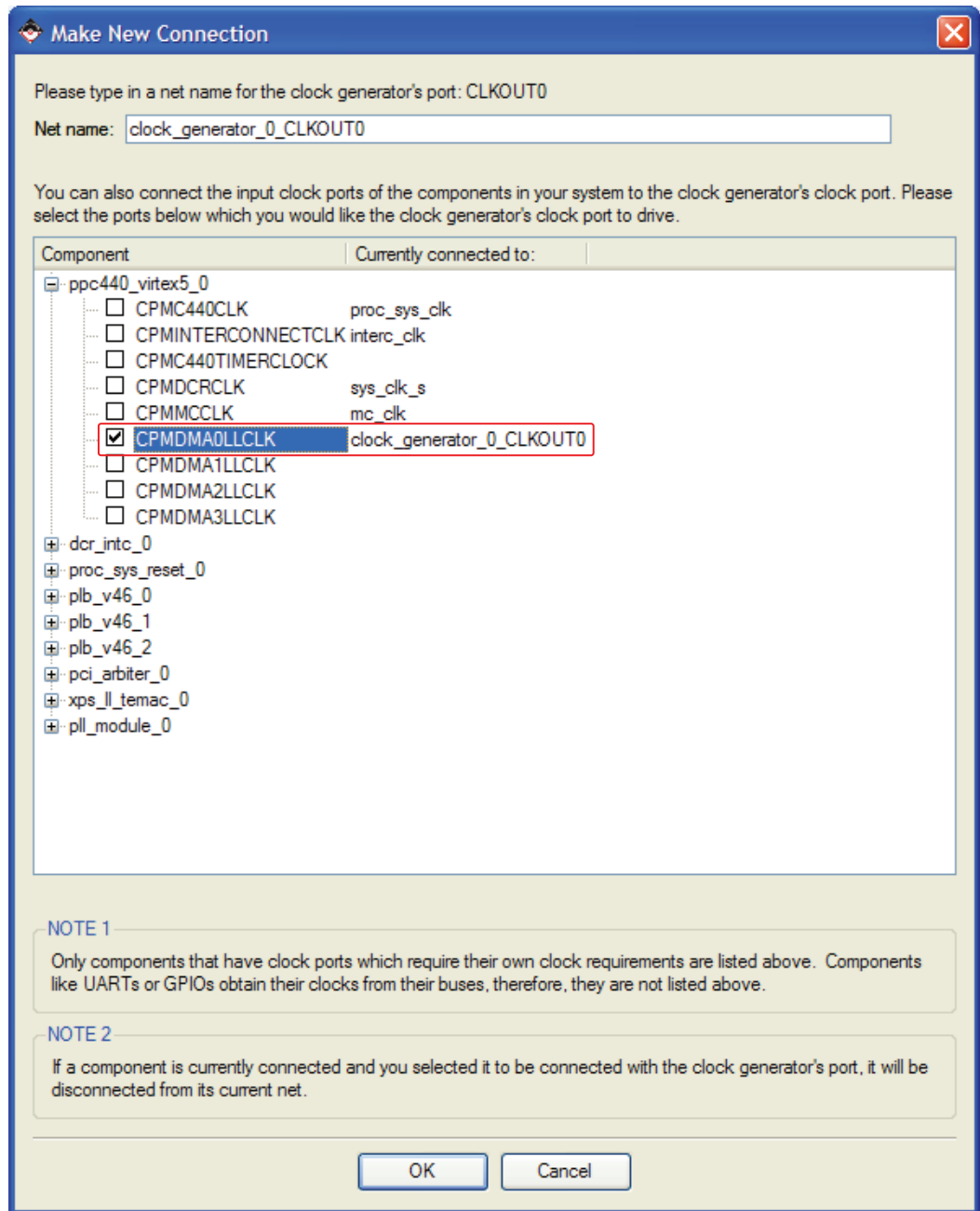
UG443_6_26_091007

Figure 2-26: Clock Generator Misc

Under the Ports section expand the Outputs tree node and click on **CLKOUT0**. On the right of the window, set the Required frequency: to **100000000 Hz**, Required phase shift : to **0**, and Grouping Information: to **NONE**. Then select **Connect to: New connection....**

In the dialog box displayed by the system, select **CPMDMA0LLCLK** under **ppc440_virtex5_0**.

The above connections are shown in [Figure 2-27](#).



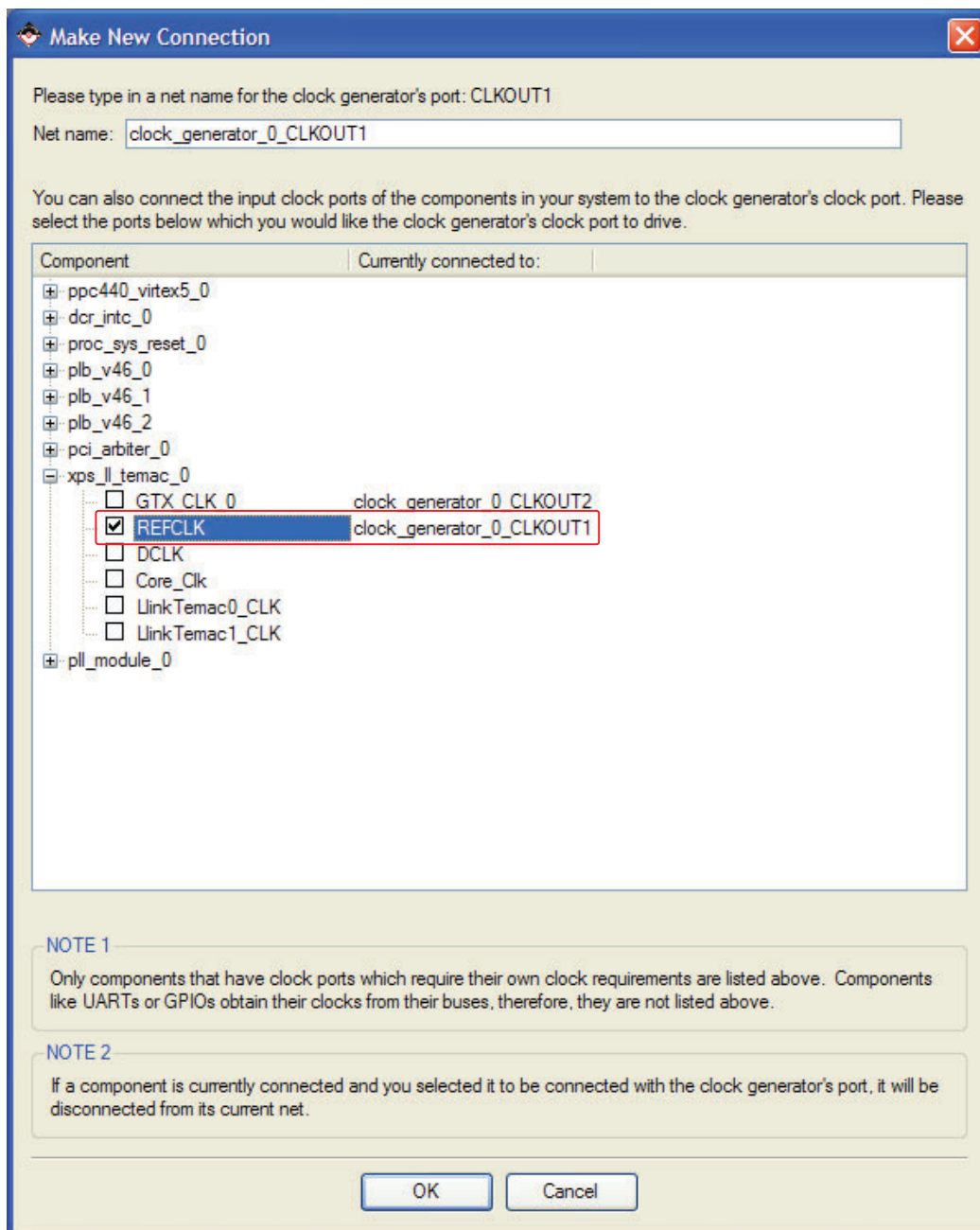
UG443_6_27_091007

Figure 2-27: Clock Generator CLKOUT0 Connections

Under Ports section, expand the Outputs tree node and click on **CLKOUT1**. On the right side of the window, set the Required frequency: to **200000000 Hz**, Required phase shift : to **0**, and Grouping Information: to **NONE**, then select **Connect to: New connection....**

Select **REF_CLK** under `xps_ll_temac_0`.

The above connections are shown in [Figure 2-28](#).



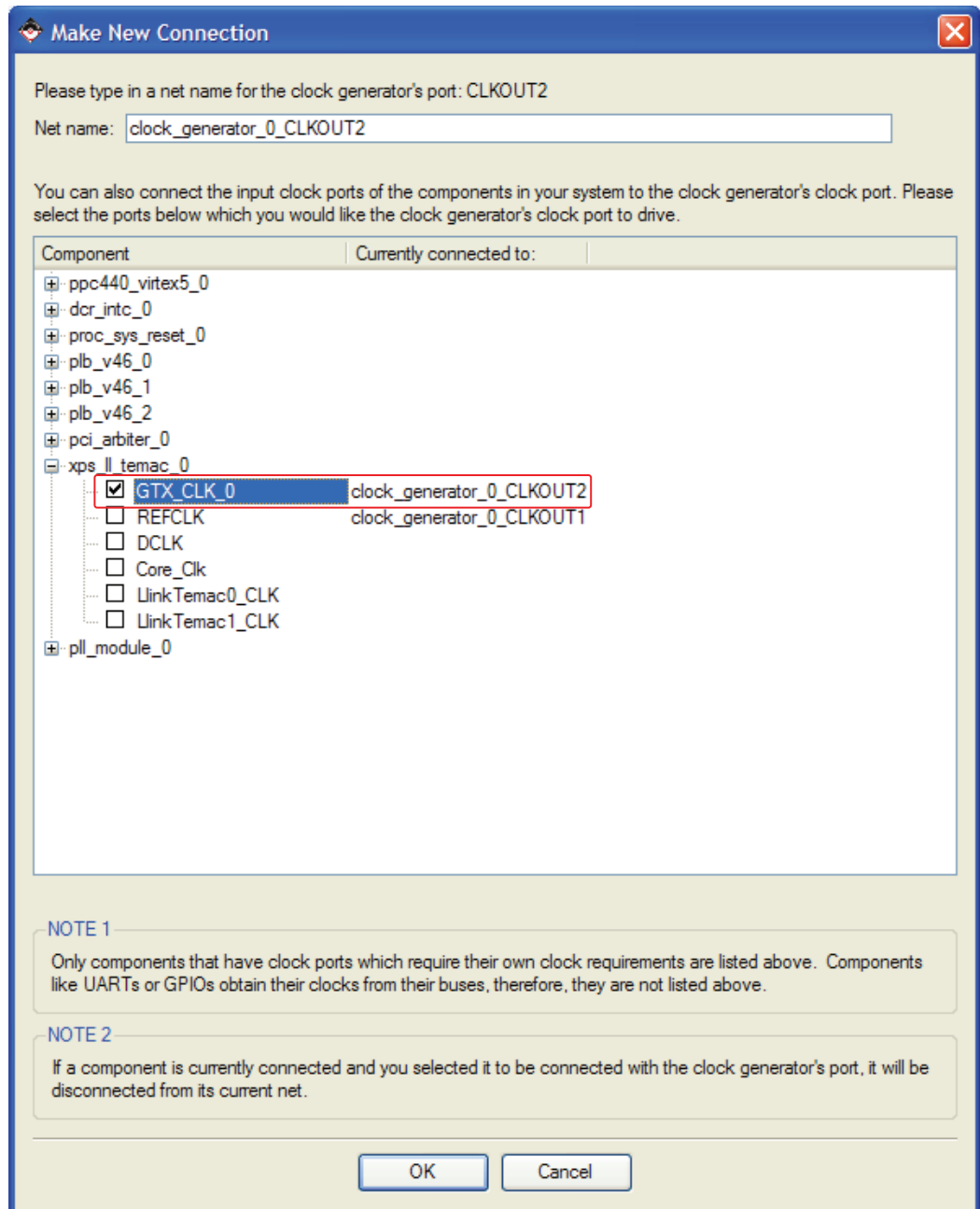
UG443_6_28_091007

Figure 2-28: Clock Generator CLKOUT1 Connections

Under the Ports section, expand the Outputs tree node and click on **CLKOUT2**. On the right of the window, set the Required frequency: to **12500000 Hz**, Required phase shift : to **0**, and Grouping Information: to **NONE**, then select **Connect to: New connection....**

Select **GTX_CLK** under **xps_ll_temac_0**.

The above connections are shown in [Figure 2-29](#).



UG443_6_29_091007

Figure 2-29: Clock Generator CLKOUT2 Connections

Migrating software from Virtex-4 PowerPC-405 to Virtex-5 PowerPC-440

The Virtex-5 processor block includes the PowerPC 440. The cross bar for the Virtex-5 is architecturally different from the Virtex-4 processor block, which included the PPC405 only. The Virtex-5 processor block keeps the software programming model changes to a minimum. In many cases the user does not have to change any code, but the Virtex-5 block offers new possibilities for system design.

Each of the following sections examines the migration requirements for one particular aspect of programming. For detailed information on using the new features of the PowerPC-440, see the *PPC440 CPU Core User's Manual*.

Instruction Set Changes

The instruction set changes primarily affect assembly language programmers. The instructions discussed here are `dcba`, `dcci`, `dcread`, `dlmzb`, `eieio`, `icbt`, `icread`, `lswx`, `lswi`, `mfspr`, `mtspr`, `stwcx`, `sync`, `tlbia`, `tlbre`, and `tlbwe`.

The PPC440 has a new, documented instruction called `dlmzb` that is useful in string operations. Although this instruction was present in some later versions of the PPC405, but was not documented. The GNU compilers supplied with EDK already use this instruction when appropriate.

The behavior of the `lswx` and `lswi` instructions is different for certain invalid instruction forms. In the PPC405, if the effective address register is in the range of registers to be written, it is skipped, while in the PPC440, it is overwritten. However, it is an error to specify the effective address register within the range of registers to be written (invalid instruction form).

The behavior of the `mfspr` and `mtspr` instructions is different for certain invalid instruction forms. In the PPC405, if the SPR address is not valid, the operation is undefined, but in the PPC440 the operation causes an invalid operation exception.

In the PPC405, the `stwcx` instruction causes an alignment exception if the effective address is not aligned on a word boundary, but an unaligned word address does not result in an exception in the PPC440. However, the normal usage of the `stwcx` instruction remains unchanged.

The `sync` and `eieio` instructions in the PPC405 are replaced by the `msync` and `mbar` instructions, respectively in the PPC440. The new instructions use the same opcodes as the old instructions so assembly code written for the PPC440 should use the new mnemonics, but the existing machine code containing these instructions will work unchanged. EDK users are insulated from these changes by the macros and functions in `xio.h`, and `xio.c`.

The `icbt` instruction has different opcodes in the PPC405 and PPC440 implementations but its behavior is the same. The `dcba` instruction in the PPC405 was intended to improve performance by providing hints to the hardware for initiating cache line fetches.

The `dcba` instruction is treated as a no-op in the PPC440 implementation. Any code that uses this instruction will continue to execute normally when moved from the PPC405 to the PPC440 but any performance advantages will be lost.

The operation of the `dcci` instruction is different between the PPC405 and PPC440 implementations. In the PPC405, `dcci` invalidates all data cache lines in a congruence class, and is intended to be invoked repeatedly to invalidate the entire cache during initialization. In the PPC440, `dcci` invalidates the entire cache, and is intended to be invoked once during initialization or power-on-reset.

The `dcread` and `icread` functions return data in a different format in the PPC440 but the instructions have the same meaning as in the PPC405.

The `tlbia` instruction in the PPC405 does not exist in the PPC440. This instruction invalidates all the TLB entries in the PPC405, and places the processor in real-address mode (as opposed to virtual-address mode). The PPC440 is always in virtual address mode so this instruction has been removed. The `tlbre`, `tlbwe`, and `tlbsx` instructions access different TLB registers in the PPC405 or TLB entry fields in the PPC440, but the meaning of the instructions remains the same.

Boot Vector

PPC440 starts at `0xFFFF_FFFC` as does the PPC405. PPC440 has a longer booting sequence because of MMU initialization and configuration requirements. The compiler and startup files provided by Xilinx handle the boot sequencing automatically.

Interrupts

In the PPC405 in Virtex-4, the result of an exception is that control is transferred to an instruction at a fixed offset from the address stored in the Exception Vector Prefix Register (EVPR). For any given exception, the predefined 16-bit offset for that exception is concatenated with the 16-bit prefix stored in the EVPR to form the address of the interrupt handler. The PPC440 in Virtex-5 improves on this mechanism by providing 16 Interrupt Vector Offset Registers (IVORs) to hold the specific offset for each exception. When an exception occurs, the 16 bit prefix stored in the Interrupt Vector Prefix Register (IVPR) is concatenated with specific bits of the corresponding IVOR to generate the address of the interrupt handler. The IVORs allow each interrupt handler to be as large or as small as needed, optimizing memory requirements. More details are available in the IBM PPC440x5 CPU Core User's Manual, Chapter 6.

The default IVOR settings generated by EDK correspond to the Virtex-4 PPC405 offsets so user code does not have to change. However, users can rewrite the code to take advantage of the IVORs, if desired.

Caches

The PPC440 in Virtex-5 has 32KB instruction and data caches, while the PPC405 in Virtex-4 had 16KB instruction and data caches. The cache organization and replacement policies are different. While these differences could affect the performance of existing user code, user code changes are not required in most cases, especially if the user code relies on access functions such as `XCACHE_EnableICache` and `XCACHE_EnableDCache` that are provided by Xilinx. These functions enable/disable 128MB chunks of memory in Virtex-4, and 258 MB chunks of memory in Virtex-5.

Users who are moving their Virtex-4 PPC405 code to Virtex-5 PPC440 should ensure that any transitions between cached and uncached regions of the memory space fall on 256 MB boundaries.

Memory Management

The PPC405 supports two memory modes – a virtual mode where the processor works with virtual addresses and the MMU translates the virtual addresses into real addresses, and a real mode where the processor works with real addresses instead of virtual addresses. The PPC440 eliminates the real mode so the processor is always in virtual

address mode. However, the real mode can be emulated by configuring the MMU and TLB appropriately.

Users of the standalone Xilinx libraries and compilers automatically start with the emulated real mode on the Virtex-5 PPC440, which is similar to the default startup mode on the Virtex-4 PPC405. The RST command in XMD puts the processor in this mode. Memory management changes from the PPC405 to the PPC440 should be transparent to users of 3rd party RTOSes.

OCM

The PPC405 in Virtex-4 has two On Chip Memory (OCM) interfaces that allowed data and instruction memory regions to be accessed directly, bypassing the cache and PLB connections. The OCM interface was designed to provide low latency access to small amounts of local memory. The OCM interface is not available in PPC440 but the behavior of the OCM can be emulated in certain cases by locking a portion of the cache.

The Virtex-5 processor block has some additional interfaces such as the Memory Controller Interface, and the Auxiliary Processor Unit Interface, that support reads or writes to a given address. However, these interfaces are not designed to be OCM replacements.

The Memory Controller Interface is designed to allow the processor to access a single high speed memory controller, but various other devices could use the crossbar to share access to this high speed memory. While the OCM interface in Virtex-4 provides low latency access to memory, the crossbar can add some cycles of latency and arbitration delays to any access from the processor to a memory connected to the Memory Controller Interface.

The APU Interface allows the processor to transfer data from its memory (cached or not-cached) or registers, to registers in a coprocessor module. This is very different from loading or storing external memory contents to/from a processor register.

Any system that was designed to use the OCM interface in Virtex-4 will therefore have to be redesigned to work without an OCM. Any memory block that was connected to the OCM interface in Virtex-4 would have to be connected to the Memory Controller Interface or to the MPLB interface in Virtex-5, along with any other memories that are required for the system.

DCR

The DCR register set for the Virtex-5 PPC440 implementation is different from the DCR register set for Virtex-4. One additional difference is that the Virtex-5 implementation provides a new indirect addressing mode, in addition to the direct addressing mode supported by the Virtex-4 implementation. The indirect addressing mode allows software to use less memory. Atomic DCR read/write operation is guaranteed by using the XIO_DCR driver.

Timers

The PPC440 implementation in Virtex-5 provides similar timer functions as the PPC405 implementation in Virtex-4. However the watchdog timer and fixed interval timer periods are different. These changes are transparent to users who access timer functions through Xilinx library functions or 3rd party RTOS functions.

APU

The APU interface and programming model are significantly different between Virtex-4 and Virtex-5. Users of the Xilinx FPU that connects to the APU interface must use a version that is compatible with the Virtex-5 APU.

Debugging

When the PPC440 operates in emulated real mode, code can be debugged using XMD to connect to the hardware, and GDB or SDK as the front end. Debugging in any other operating mode is not available through XMD for the Beta release, but Linux and Vxworks systems can be debugged with the help of third-party debuggers and hardware such as the BDI2000 debug module.

PPC405 - PPC405/PLB v4.6 System Migration

Introduction

This chapter describes the system migration from a PPC405 PLB v3.4 system to a PPC405 PLB v4.6/XPS system. The migrated PPC405 PLB v4.6/XPS system uses the Multi-Ported Memory Controller (MPMC) core for main memories like SDRAM, DDR, and DDR2.

The following high-level migration steps are discussed:

- PPC405 wrapper with PLB v4.6 interface
- PLB v4.6 instances are added to the system where the OPB/PLB v3.4 and bridges are removed
- Clocking/Reset scheme
- Memory controllers are migrated to MPMC which is integrated into system
- Ethernet solution
- Equivalent XPS cores (from OPB/PLB v3.4) are connected to PLB v4.6

The original and migrated systems are built for the ML403 board.

System Migration Methodology

System Hardware Migration

The method recommended flow for doing the system hardware migration is outlined below.

- Migrate the processor from the PPC405 to the PPC405 with a PLB v4.6 interface
- Add PLB v4.6 instances for the system
- Migrate the PLB v3.4 or OPB memory controller to MPMC
- Migrate the clocking/reset scheme
- For peripherals, Master and Slaves cores are connected to the PLB v4.6
 - ◆ The maximum number of masters that can be connected to the SPLB is 16
 - ◆ The maximum number of slaves that can be connected to MPLB is 16

System Software Migration

No modifications to user application code are necessary when migrating from a PPC405 system to a PC405 PLB v4.6 system.

Existing PPC405 PLB v3.4 System

An existing PPC405 PLB v3.4 system is shown in [Figure 3-1](#). This system will be used as a migration example since it contains common cores that are found inside a typical embedded system.

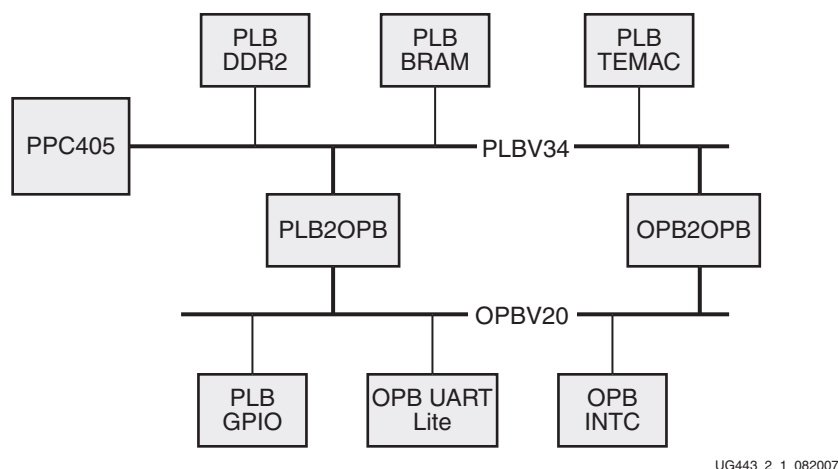


Figure 3-1: PPC405\PLBV34 System

PPC405 PLB v4.6/XPS System

The migrated PPC405 PLB v4.6/XPS system is shown in [Figure 3-2](#).

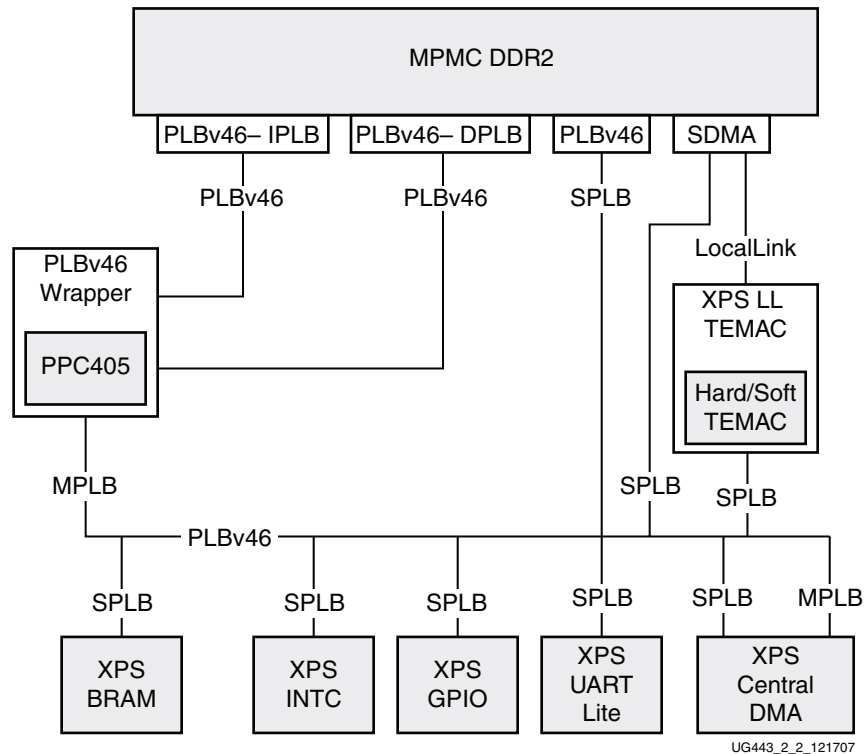


Figure 3-2: Migrated PPC405 PLB v4.6 System

Overview of System Migration

Processor Migration

The system's PPC405 Virtex-4 (Wrapper) must use v2_00_a which supports the PLB v4.6 interface.

Added features of this wrapper are dual instruction-side and dual data-side PLB v4.6 interfaces.

The IPLB1 and DPLB1 bus interfaces are used for Point-to-Point connections to the PLB v4.6 PIMs on the MPMC.

The IPLB0 and DPLB0 bus interfaces are both connected to the PLB v4.6 shared bus. The shared bus is connected to a PLB v4.6 PIM port on the MPMC.

Note: The PLB v4.6 PIM port on the MPMC for the shared bus is only required when a master is connected to the shared bus and needs to access the MPMC address range.

A block diagram is shown in [Figure 3-3](#).

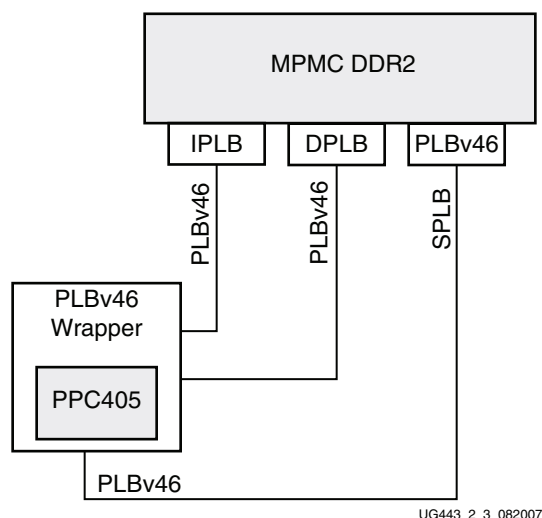


Figure 3-3: PPC405 Bus Interfaces Block Diagram

Interconnect

The PLB v3.4 and OPB instances are deleted and the PLB v4.6 instance is added for the shared bus. In addition, the PLB2OPB and OPB2PLB bridges are removed. PLB v4.6 instances are added for the IPLB1 and DPLB1 and the shared buses IPLB0 and DPLB0.

XPS Peripherals

This section covers the general migration from an OPB/PLB v3.4 inside the system to a XPS core. This is applicable to slave and master/slave cores.

1. The former PLB v3.4 or OPB core is deleted.
2. The XPS core is added to the system.
3. Parameters and ports are connected in the same manner as the PLB v3.4 or OPB equivalent.
4. The slave or master/slave interfaces are connected to the PLB v4.6 shared bus instance.

Memory Controller Migration

The memory controller for SDRAM, DDR and DDR2 memories in a PLB v4.6 system is the MPMC.

The MPMC can support up to 8 configurable ports. Ports can be configured as XCL, PLB v4.6, SDMA (Soft Direct Memory Access) and NPI (Native Port Interface).

In the example migrated system block diagram, the MPMC is configured to support 4 ports and uses the PLB v4.6 and the SDMA (soft Direct Memory Access) Port Interface Modules (PIMs).

The PLB v4.6 PIMs are used for ports to the IPLB1 (Port 0) and DPLB1 (Port 1) on the PPC405 processor. The PLB v4.6 PIM contains a slave interface. Port 2 allows other masters connected to the PLB v4.6 shared bus to have access to the main memory.

Note: The PLB v4.6 PIM port on the MPMC for the shared bus is only required when a master is connected to the shared bus and needs to access the MPMC address range.

The SDMA PIM is connected to Port 3.

The SDMA PIM LocalLink interface is connected to the XPS LL TEMAC core to provide DMA accesses in and out of memory. The SDMA PIM is connected to the PLB v4.6 bus as a slave and to the XPS LL Temac through a LocalLink interface. The PLB v4.6 slave connection allows masters on the PLB v4.6 bus to setup and control DMA operations.

Ethernet Controller Migration

The migration of the PLB TEMAC to the XPS LL TEMAC controller varies depending on the features configured for the PLB TEMAC core.

If the PLB TEMAC core in the PPC405 PLB v3.4 system was configured to include the Scatter-gather DMA feature, the migrated system will require a DMA Controller with Scatter-gather DMA capability to be added. The SDMA PIM provides Scatter-gather DMA capability to and from the MPMC. The XPS LL TEMAC connects to this DMA controller via the LocalLink port.

If the PLB TEMAC core in the PPC405 PLB v3.4 system was configured to use FIFO direct mode, the migrated system will require the addition of the XPS LL FIFO core. The XPS LL TEMAC connects to the XPS LL FIFO via the LocalLink port. The FIFO data is then accessed through the PLB v4.6 connection of the XPS LL FIFO.

XPS LL TEMAC instantiates either a hard TEMAC or a soft TEMAC into the core. For a V4FX system, XPS LL TEMAC will instantiate the hard TEMAC. XPS LL TEMAC has two hard TEMACs available inside the core.

System Migration Inside an EDK System

Migrating Buses/Processor

Migrating the Bus

Adding/Removing Bus Instances

Remove the `opb`, `plb` and `plb2opb` instances (Delete instance and its internal ports) inside the System Assembly View/Bus Interfaces.

In addition, add 3 PLB v4.6 instances by expanding the Bus Bridge tree node. Right click on **Processor Local Bus (PLB) 4.6** and select **Add IP**. The buses are for the `IPLB1`, `DPLB1`, `IPLB0` and `DPLB0`. The instances created are `plbv46_0`, `plbv46_1`, and `plb_v46_2`.

Migrating the Processor

Adding/Removing PPC405 Instances

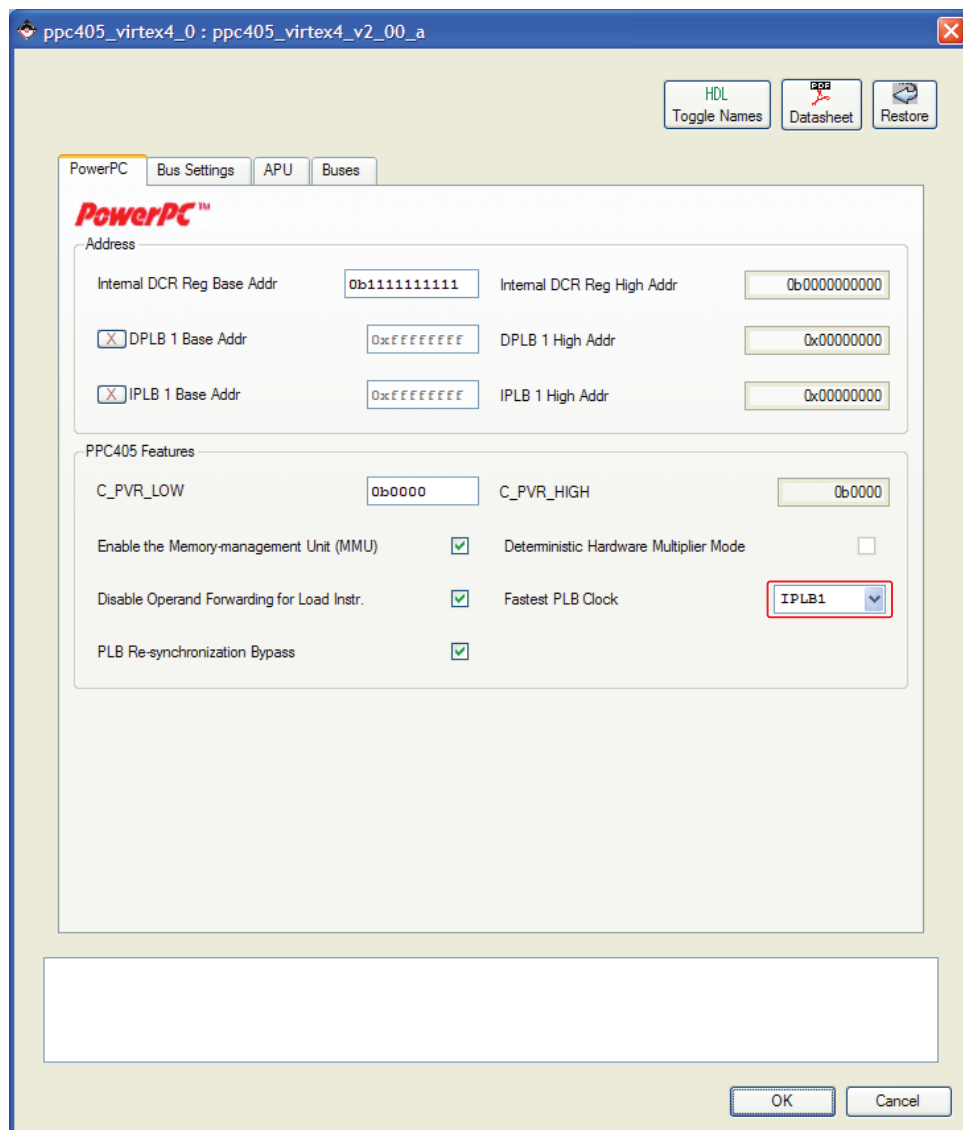
Remove the former `ppc405_0` instance by right clicking on `ppc405_0` inside the **System Assembly View/Bus Interfaces** and clicking on **Delete Instance...** (Delete instance and its internal ports).

Add the latest version of the processor by expanding the Processor tree node inside the IP Catalog tab. Right click on **PowerPC 405 Virtex-4** and click on **Add IP**. This creates the `ppc405_virtex4_0` instance.

Setting Parameters

Right click on ppc405_virtex4_0 inside the System Assembly View and select **Configure IP** ...

Inside the PowerPC tab select the **Fastest PLB Clock** which is **IPLB1** as shown in [Figure 3-4](#).

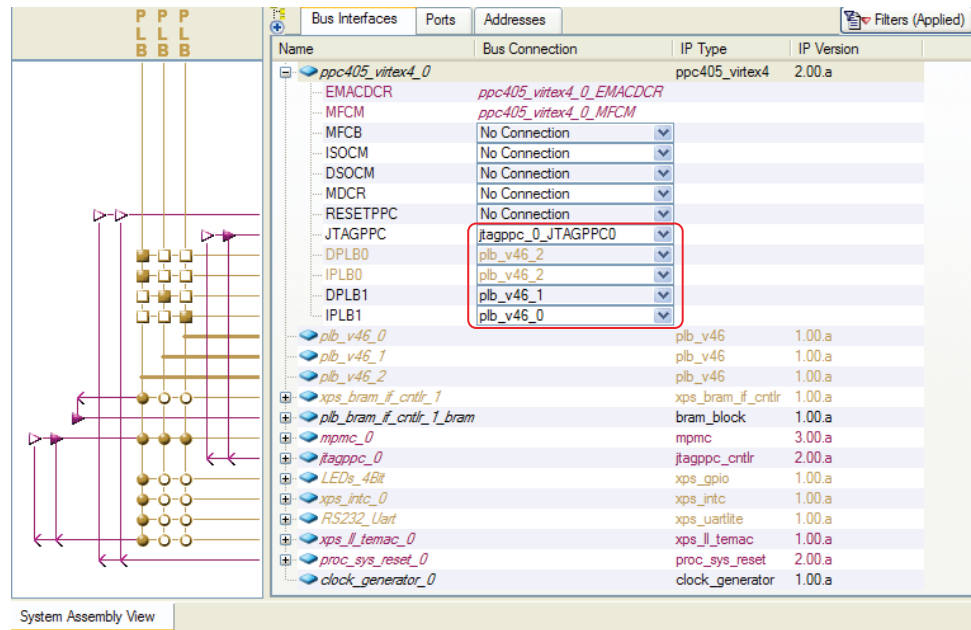


UG443_2-4_082007

Figure 3-4: Setting PPC405 Parameters

Setting Bus Interfaces

Inside the Bus Interfaces Tab, expand the ppc405_virtex4_0 instance. Select the following bus interfaces, JTAGPPC bus connection is connected to jtagppc_0_JTAGPPC0, IPLB1 bus is connected to plb_v46_0, DPLB1 bus is connected to plb_v46_1, IPLB0 and DPLB0 bus connections are plb_v46_2. The above selections are shown in Figure 3-5.



UG443_2_5_082007

Figure 3-5: Setting PPC405 Bus Interfaces

Migrating The Memory Controller

Adding/Removing Memory Controllers

Remove the **DDR_SDRAM_64Mx32** instance by right clicking on **DDR_SDRAM_64Mx32** inside the **System Assembly View/Bus Interfaces** and clicking on **Delete Instance...** Then click on **Delete instance and its internal ports**.

Add MPMC to the system by expanding the Memory Controller tree node inside the IP Catalog tab. Right click on **Multi-Port Memory Controller** and click on **Add IP**. This creates the `mpmc_0` instance.

The PLB DDR memory controller provided a big endian connection to memory devices whereas MPMC provides a little endian connection to memory devices. In turn the user needs to swap the pins to match the proper endian of the MPMC. In addition, it is preferred to use a MIG complaint pinout to obtain the best performance with MPMC.

Refer to the MPMC documentation for IOSTANDARDS required for certain pins for the UCF which might be different from the previous memory controller.

Configuring The MPMC

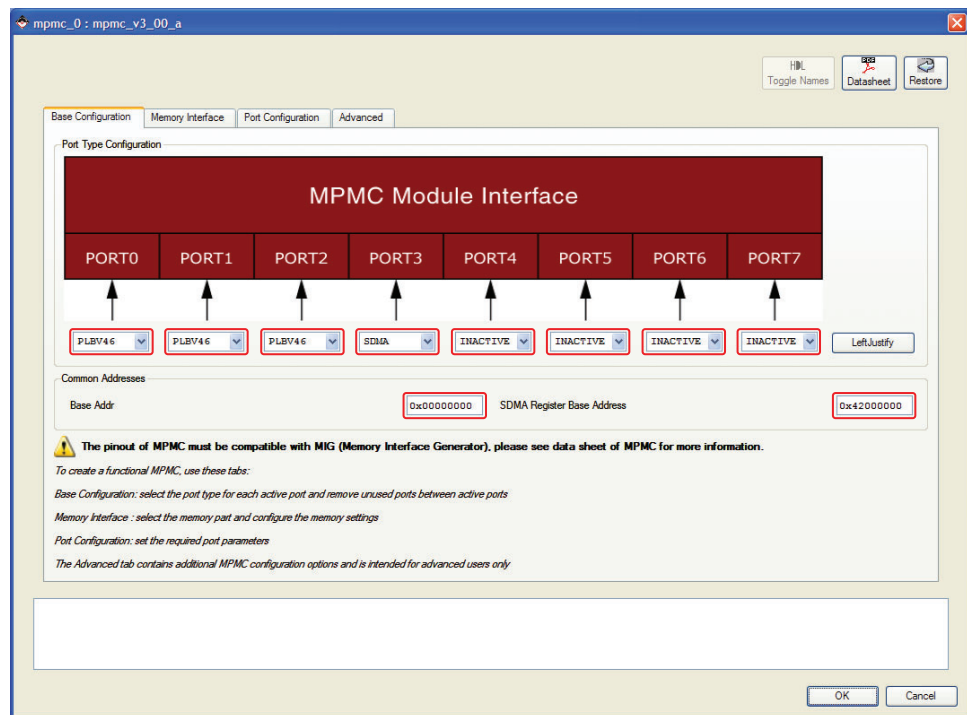
Right click on mpmc_0 inside the System Assembly View and select **Configure IP ...**.

Setting Ports

Inside the **Base Configuration** Tab, four ports will be used on the MPMC. Port 0 is set to the **Port Type** of **PLBV46**. Port 1 is set to the **Port Type** of **PLBV46**. Port 2 is set to the **Port Type** of **PLBV46**. Port 3 is set to the **Port Type** of **SDMA**.

In addition, under Common Addresses, the **Base Addr** is set to **0x00000000** which is the base address of the PLB DDR and the **SDMA Register Base Address** is set to **0x42000000** in this case. All ports of the MPMC are sharing the same base and high address.

The above configuration of the ports is shown in [Figure 3-6](#).



UG443_2_6_082007

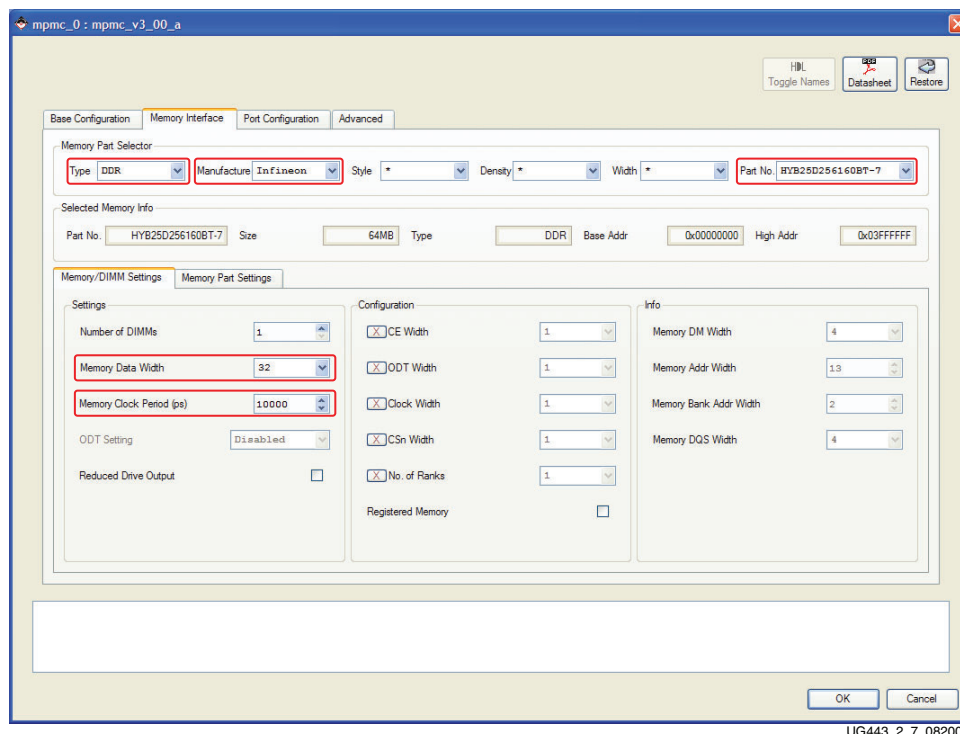
Figure 3-6: MPMC Base Configuration Tab

Inside the **Memory Interface** Tab, selections are made for the DDR memory on the board.

For Memory Part Filter, **Type** is set to **DDR**, **Manufacture** is set to **Infineon**, and **Select a Part** is set to **HYB25D256160BT-7**. The Part Settings, Memory Timing Information and DIMM Settings sections are set automatically based upon the DDR selected inside the Memory Part Selector section.

For the Memory System Settings section, **Memory Clock Period** is set to **10000ps**, and **Memory Data Width** is set to **32**.

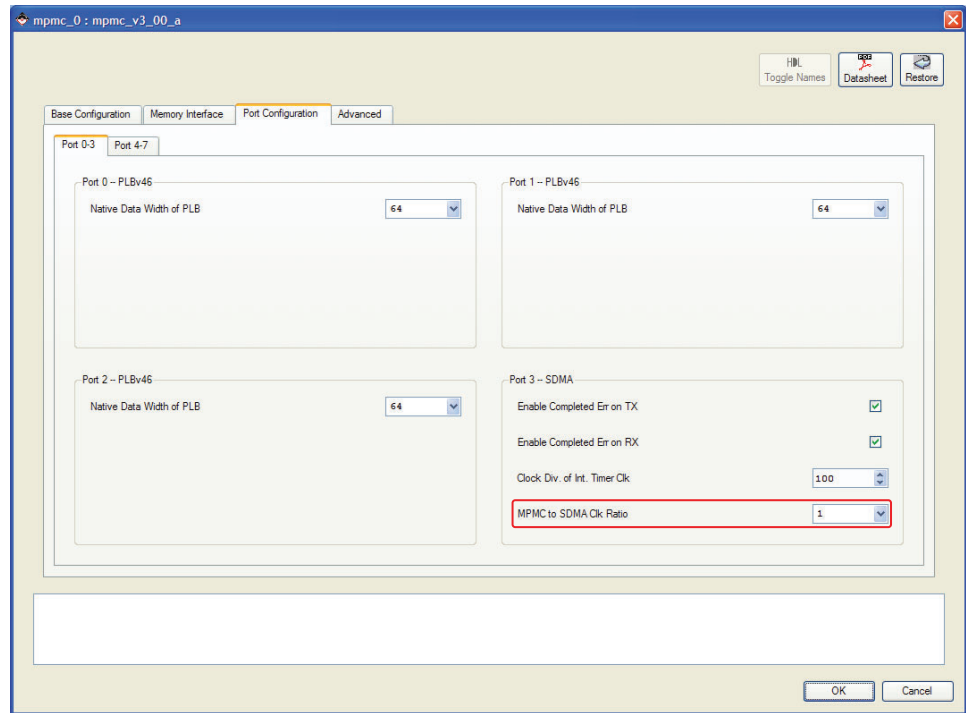
The above selections are shown [Figure 3-7](#).



UG443_2_7_082007

Figure 3-7: MPMC Memory Interface Tab

Inside the Port Configuration Tab, Port 3- SDMA, MPMC to SDMA Clk Ratio is set to 1 as shown in [Figure 3-8](#).



UG443_2_8_082007

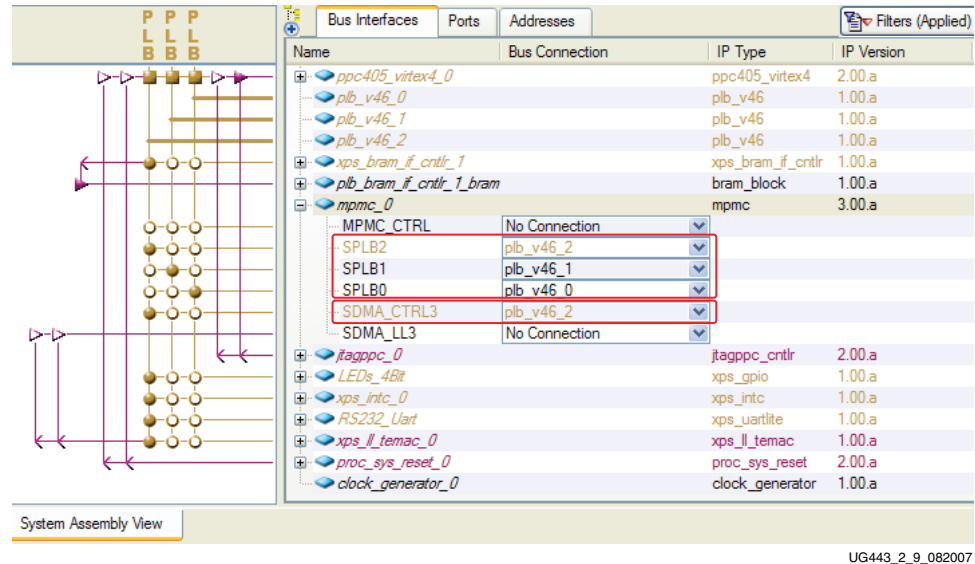
Figure 3-8: MPMC Port Configuration Tab

Inside the Advanced/Misc Tab, set the `C_NUM_IDELAYCTRL` to 2 and `C_IDELAYCTRL_LOC` to `IDELAYCTRL_X0Y2-IDELAYCTRL_X0Y3`. These settings are for the ML403 board.

MPMC Bus Interfaces

Ports 0,1, and 2 of the MPMC contains PLB v4.6 PIMs. The **SPLB0**, **SPLB1** and **SPLB2** bus interfaces are connected to the **plb_v46_0**, **plb_v46_1**, and **plb_v46_2** instances respectively as shown in [Figure 3-9](#).

In addition, Port 3 is configured for SDMA. The **SDMA_CTRL3** bus interface is connected to **plb_v46_2** which is slave interface on the SDMA. This is shown in [Figure 3-9](#).



UG443_2_9_082007

Figure 3-9: MPMC PLBV46 PIM Bus Interfaces

MPMC Ports

The MPMC DDR ports are connected similar to the PLB DDR ports. The external ports for DQ and DQS external ports the Name and Net name must be the same and the external DDR feedback clock is removed. Discussion of connecting the system clocking needed for the memory controller is discussed later in this chapter.

In addition, the SDMA TX and RX interrupts should be connected to interrupt controller inside the system.

Migrating Ethernet Solution

Adding/Removing Ethernet

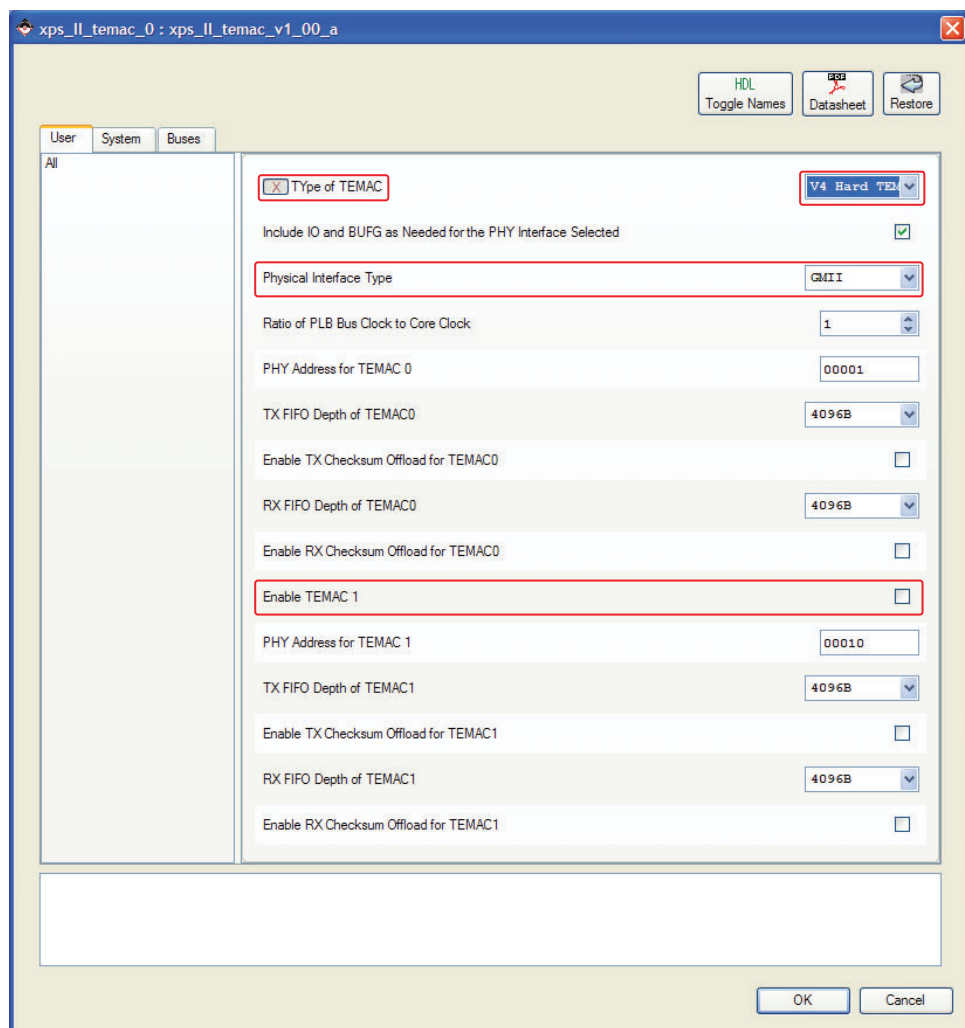
Remove the **TriMode_MAC_GMII** instance by right clicking on **TriMode_MAC_GMII** inside the **System Assembly View/Bus Interfaces** and clicking on **Delete Instance...** . Then click on **Delete instance and its internal ports**.

Add XPS LL TEMAC to the system by expanding the Communication High-Speed tree node inside the IP Catalog tab. Right click on **XPS LocalLink Tri-mode Ethernet Mac** and click on **Add IP**. This creates the `xps_ll_temac_0` instance.

Setting Parameters

Right click on `xps_ll_temac_0` inside the System Assembly View and select **Configure IP ...** .

The XPS LL TEMAC is configured to use a single Hard TEMAC by unselecting **Enable TEMAC1**. The GMII interface is used by setting **Physical Interface Type** to **GMII**. Since the system is for Virtex™-4, the **Type of TEMAC** is set to **V4 Hard TEMAC**. The above selections are shown in [Figure 3-10](#).



UG443_2_10_082007

Figure 3-10: XPS LL TEMAC Parameters

Setting Bus Interfaces

The slave interface of the XPS LL TEMAC core is connected to the plb_v46_2 instance inside the system, as shown in Figure 3-11.

Port 3 of the MPMC contains the SDMA. The LocalLink bus interface from the XPS LL TEMAC is connected to the SDMA LocalLink bus interface on the MPMC, as shown in Figure 3-11.

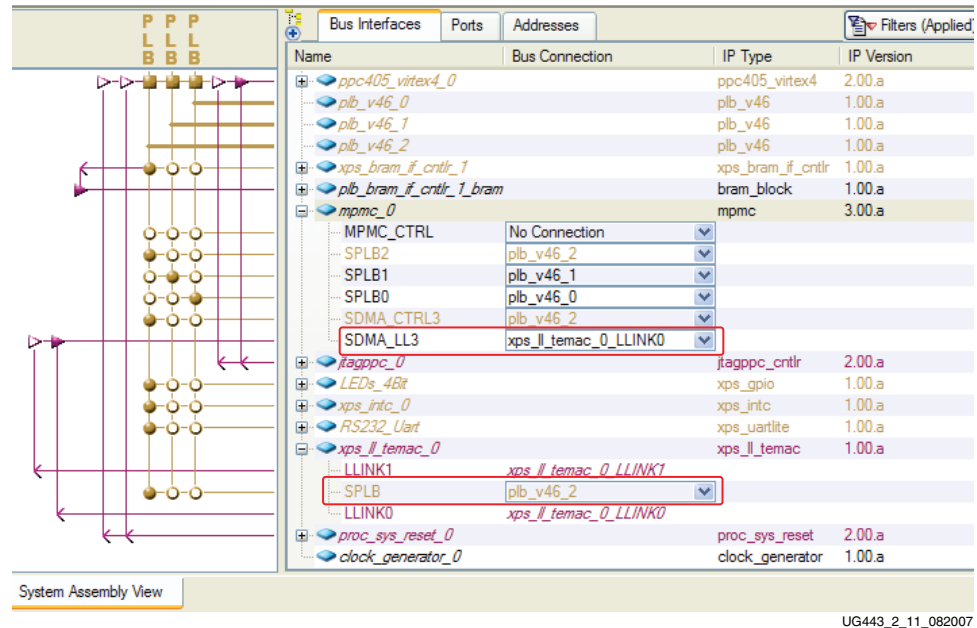


Figure 3-11: XPS LL TEMAC Bus and LocalLink Interfaces

XPS LL TEMAC Ports

The XPS LL TEMAC GMII ports are connected similar to PLB TEMAC GMII ports. Discussion of connecting the system clocking for the XPS LL TEMAC is discussed later in this chapter.

Migrating XPS Peripherals

Inside the migrated system, the PLB BRAM, OPB INTC, OPB GPIO, and OPB UART Lite cores are replaced with the equivalent XPS BRAM, XPS INTC, XPS GPIO, and XPS UART Lite, respectively.

Expand the tree node for `ppc405_virtex4_0`. The `EICC405EXTINPUTIRQ` input signal should be connected to the `IRQ` output signal on the interrupt controller inside the system.

Adding these cores are not discussed.

Modifying Clocking/Reset Inside The System

The processor system reset core is replaced with v2_00_a version of the core. In addition, the DCMs instantiations inside the system are replaced with the clock generator core.

Processor System Reset

Adding Processor System Reset to Migrated System

Inside the System Assembly View/Bus Interfaces, delete the reset_block instance (Delete instance and its internal ports).

Add the latest version of the processor system reset by expanding the Clock Reset and Interrupt tree node inside the IP Catalog tab. Right click on **Processor System Reset Module** and click on **Add IP**. This creates the proc_sys_reset_0 instance.

Right click on proc_sys_reset_0 inside the System Assembly View and select **Configure IP**

Set **External Active High** to 0, Number of **Bus Structure Reset Registered Outputs** set to 4. The above parameters are shown in [Figure 3-12](#).

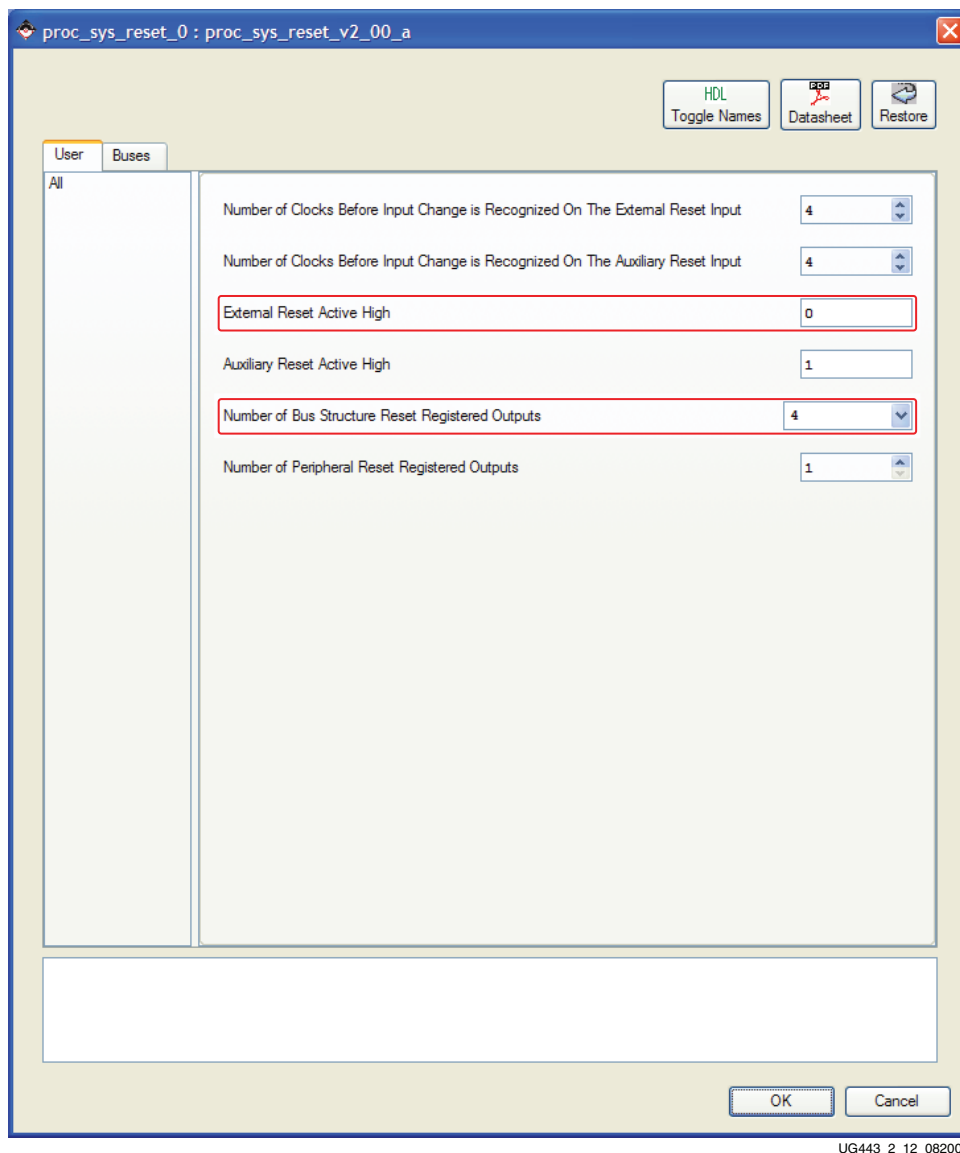
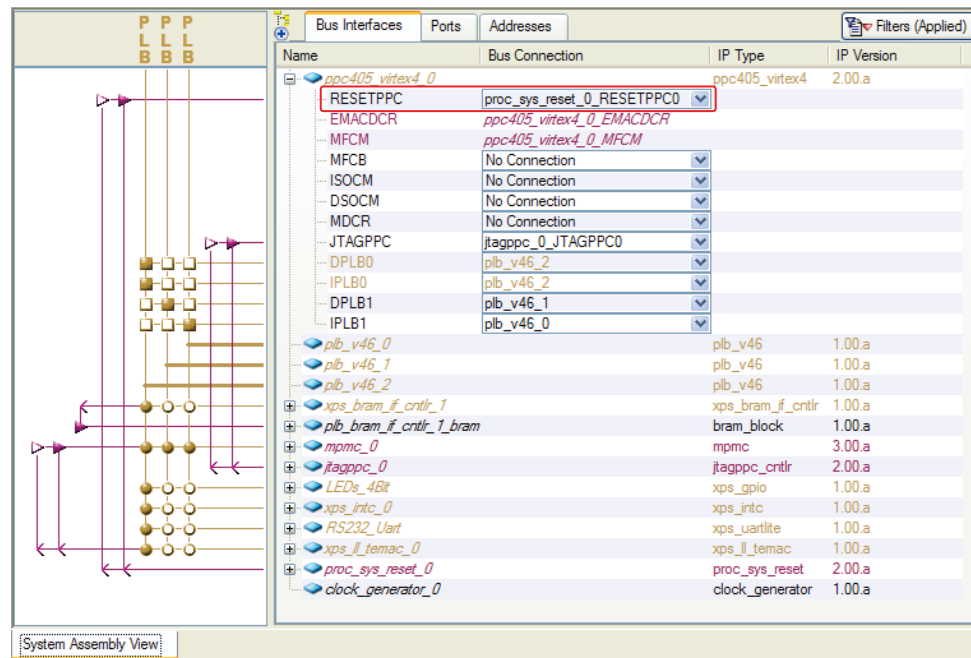


Figure 3-12: Processor System Reset Parameters

Inside the Bus Interfaces Tab, expand the ppc405_virtex4_0 tree node. Connect RESETPPC to proc_sys_reset_0_RESETPPC0 as shown in Figure 3-13.

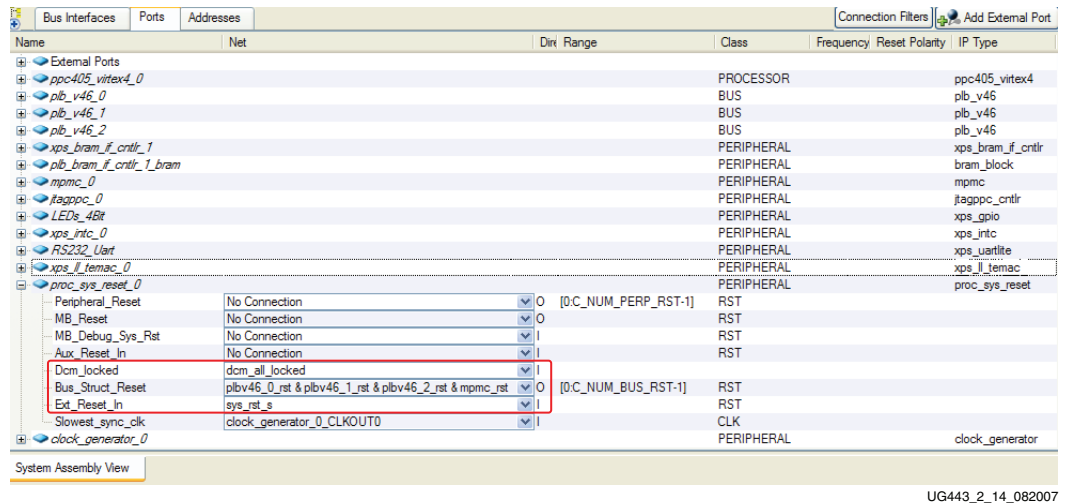


UG443_2_13_082007

Figure 3-13: PPC405 Reset Bus Interface

Inside the Ports Tab, expand the tree node for `proc_sys_reset_0`. Set `Ext_Reset_in` to `sys_rst_s`. `Bus_Struct_Reset` to `plbv46_0_rst`, `plbv46_1_rst`, `plbv46_2_rst`, and `mPMC_rst`. In addition, set `Dcm_locked` to `dcm_all_locked` which the input signal is created inside the clocking section.

The above port connections are shown in [Figure 3-14](#).



Name	Net	Dir	Range	Class	Frequency	Reset Polarity	IP Type
External Ports							
ppc405_virtex4_0				PROCESSOR			ppc405_virtex4
plb_v46_0				BUS			plb_v46
plb_v46_1				BUS			plb_v46
plb_v46_2				BUS			plb_v46
xps_bram_if_ctrlr_1				PERIPHERAL			xps_bram_if_ctrlr
plb_bram_if_ctrlr_1_bram				PERIPHERAL			bram_block
mPMC_0				PERIPHERAL			mPMC
flagppc_0				PERIPHERAL			flagppc_ctrlr
LEDs_4Bt				PERIPHERAL			xps_gpio
xps_intc_0				PERIPHERAL			xps_intc
RS232_Uart				PERIPHERAL			xps_uartlite
xps_ll_femac_0				PERIPHERAL			xps_ll_femac
proc_sys_reset_0				PERIPHERAL			proc_sys_reset
Peripheral_Reset	No Connection	0	[0:C_NUM_PERP_RST-1]	RST			
MB_Reset	No Connection	0		RST			
MB_Debug_Sys_Rst	No Connection	1		RST			
Aux_Reset_In	No Connection	1		RST			
Dcm_locked	dcm_all_locked	1					
Bus_Struct_Reset	plbv46_0_rst & plbv46_1_rst & plbv46_2_rst & mPMC_rst	0	[0:C_NUM_BUS_RST-1]	RST			
Ext_Reset_In	sys_rst_s	1		RST			
Slowest_sync_clk	clock_generator_0_CLKOUT0	1		CLK			
clock_generator_0				PERIPHERAL			clock_generator

Figure 3-14: Processor System Reset Connections

Expand the tree nodes for `plb_v46_0`, `plb_v46_1`, and `plb_v46_2` inside the System Assembly View/Port. Connect the `SYS_Rst` in all bus instances to `plbv46_0_rst`, `plbv46_1_rst`, and `plbv46_2_rst` respectively.

For master and slave peripherals connected to the PLB v4.6, the peripherals get vectorized resets for each master and slave connection from the bus core.

Expand the tree node for `mPMC_0` and set the `MPMC_Rst` port to `mPMC_rst`.

Clock Generator

Overview of Existing Clocking Scheme

Originally, the system contains 2 cascading DCMs. The first DCM provides the 100 MHz clocks for the bus, processor clock, and the PLB DDR CLK0. In addition, the first DCM provides the 125 MHz clock for the Hard TEMAC.

The second DCM provides the 100 MHz phase shifted CLK90 for the PLB DDR memory controller.

With the PLB DDR clocks, inverters are needed for the CLK0 and CLK90 from the DCMs to provide the PLB DDR CLK180 and CLK270 clocks.

Clocking Requirements of Migrated System

The migrated system requires a 100 MHz clock for the processor, and buses. In addition, MPMC needs a 200 MHz clock for the IDELAY Controllers and a 100 MHz CLK0 and CLK90 clocks. No additional phase shift is needed for CLK90 since calibration for MPMC is handled by the IDELAY Controllers. Also, a 125 MHz clock is needed for the Hard TEMAC.

Configuring Clock Generator Inside The Migrated System

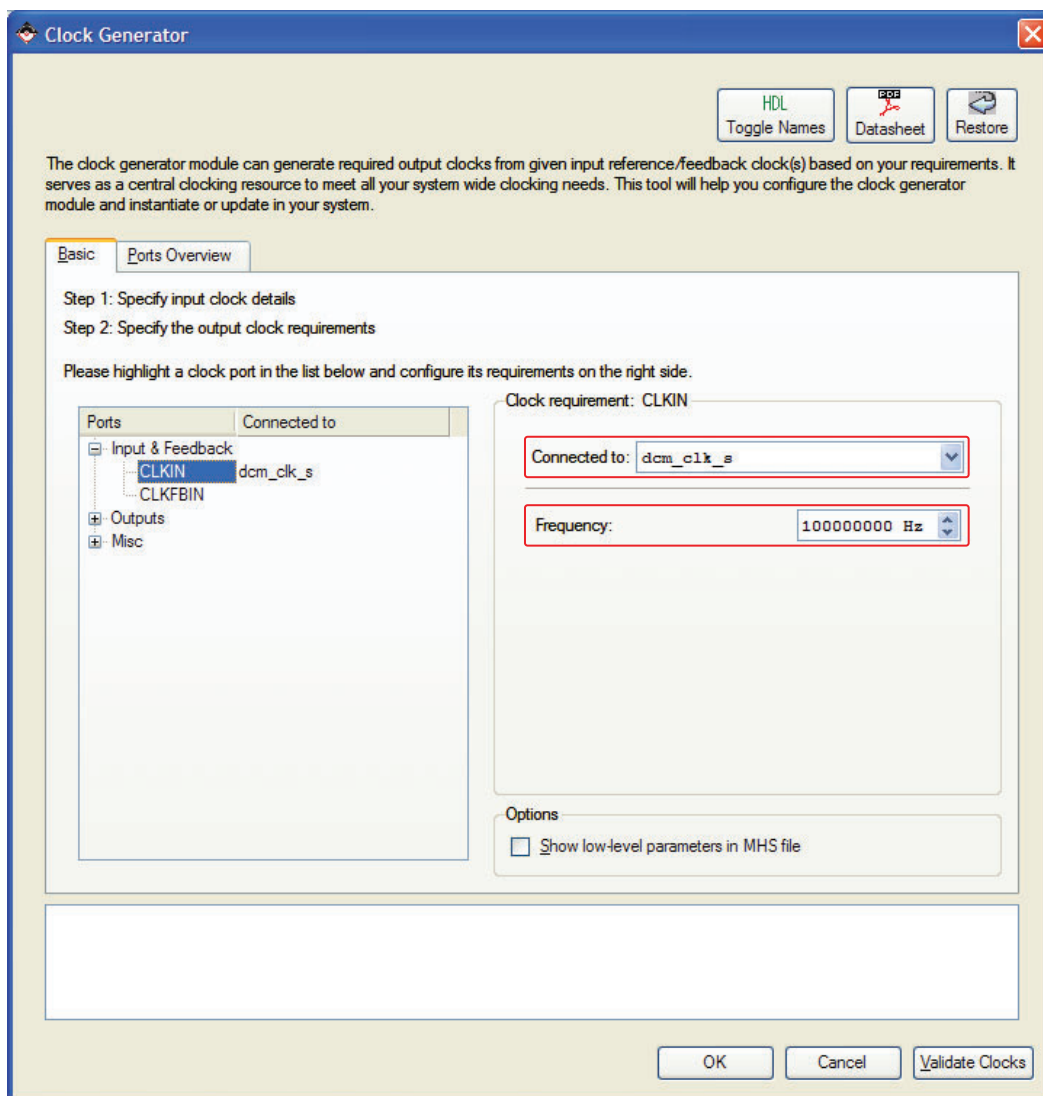
Inside the System Assembly View/Bus Interfaces, delete the dcm_0 and dcm_1 instances (Delete instance and its internal ports).

Add the clock generator by expanding the **Clock Reset and Interrupt** tree node inside the IP Catalog tab. Right click on **Clock Generator** and click on **Add IP**. This creates the clock_generator_0 instance inside the system.

Right click on clock_generator_0 inside the System Assembly View and select **Configure IP ...**.

Inside the Basic tab is where the user sets up the inputs and outputs for the Clock Generator.

Under the Ports, expand the Input & Feedback tree node. Click on CLKIN. On the right side of the window, select **Connect to:** `dcm_clk_s`. In addition, change the **Frequency:** to 100000000 Hz as shown in Figure 3-15.



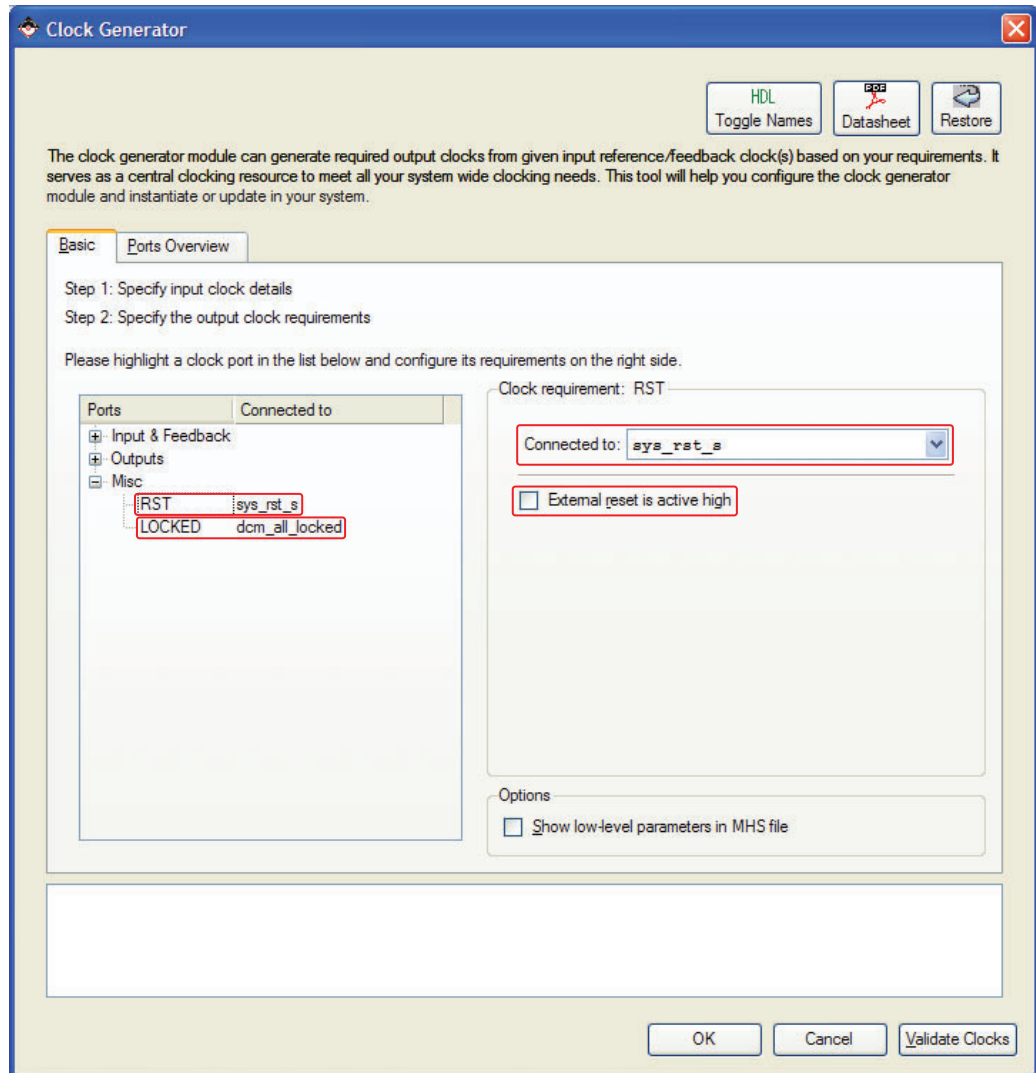
UG443_2_15_082007

Figure 3-15: Clock Generator Input and Feedback

Under the Ports, expand the Misc tree node which is near the bottom. Click on **RST**. On the right side of the window, select **Connect to: sys_rst_s**. Don't select External reset is active high since the polarity on the reset is active low on the board.

Click on **LOCKED**. On the right side of the window, select **Connect to: dcm_all_locked**. This output is connected to an input on the processor system reset module.

The above connections are shown in [Figure 3-16](#).



UG443_2_16_082007

Figure 3-16: Clock Generator Misc

Under the Ports expand the Outputs tree node and click on CLKOUT0. On the right of the window, set the **Required frequency:** to 100000000 Hz, **Required phase shift :** to 0, and **Grouping Information:** to Group0. Then select **Connect to: New connection...** .

This brings up a dialog box.

Select **Slowest_sync_clk** under **proc_sys_reset_0**.

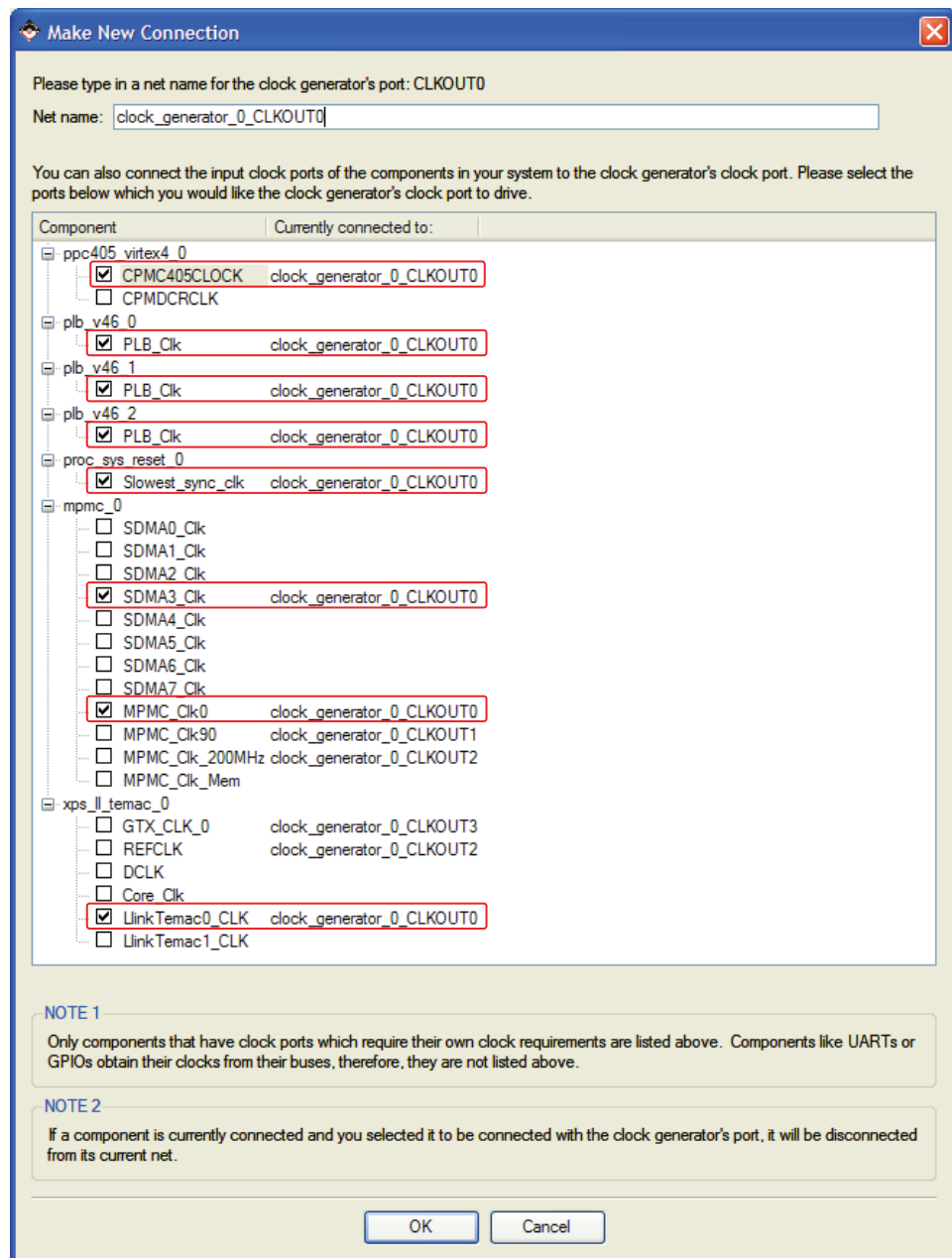
Select **PLB_Clk** under **plb_v46_0**, **plb_v46_1**, and **plb_v46_2**.

Select **CPMC405CLOCK** under **ppc405_virtex4_0**.

Select **SDMA3_Clk** and **MPMC_Clk0** under **mpmc_0**. This sets the SDMA clock and CLK0 of the MPMC.

Select **LlinkTemac0_CLK** under **xps_ll_temac_0** which selects the LocalLink clock.

The above connections are shown in [Figure 3-17](#).



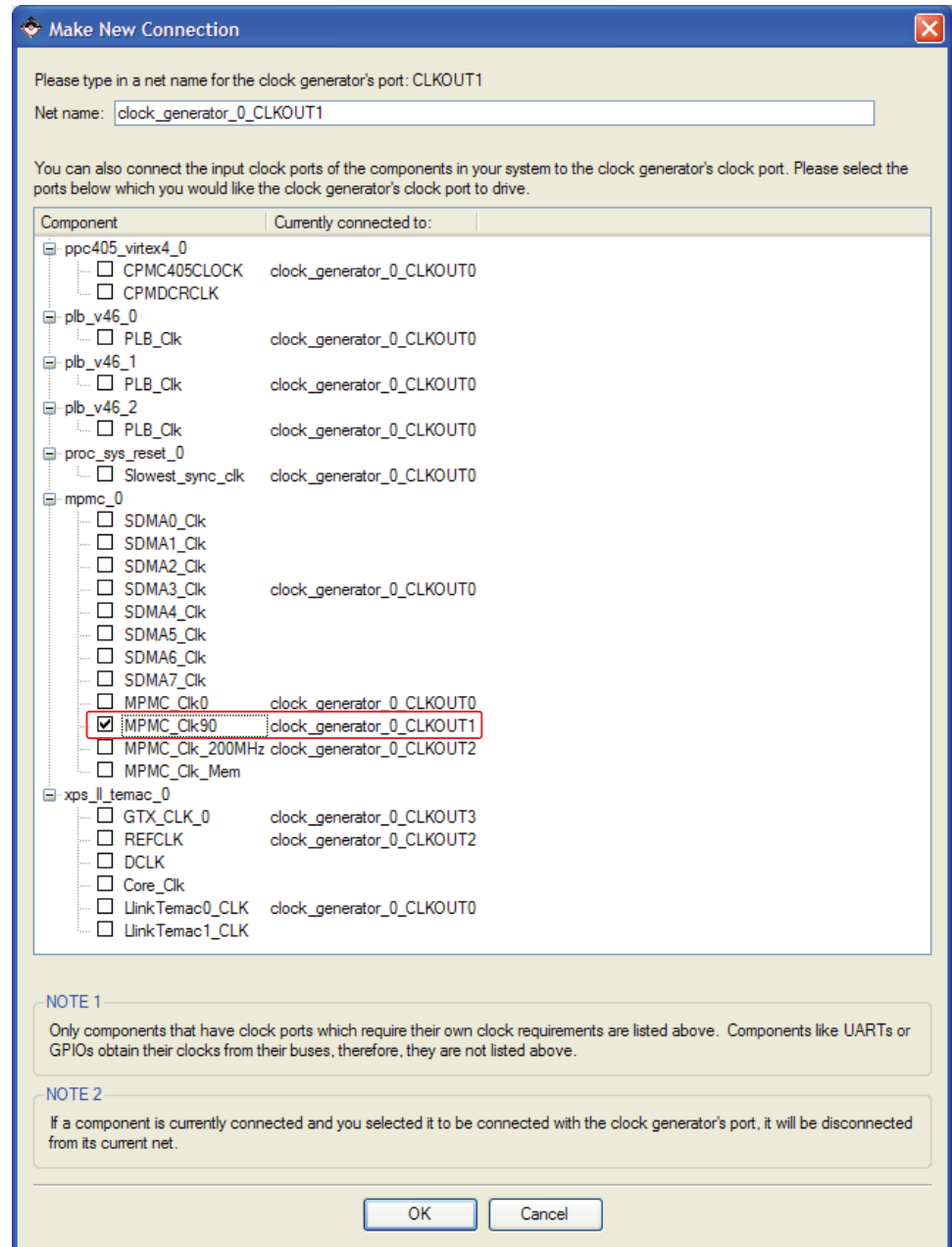
UG443_2_17_082007

Figure 3-17: Clock Generator CLKOUT0 Connections

Under the Ports expand the Outputs tree node and click on CLKOUT1. On the right of the window, set the **Required frequency:** to 100000000 Hz, **Required phase shift :** to 90, and **Grouping Information:** to Group0. Then select **Connect to: New connection...**

Select **MPMC_Clk90** under **mpmc_0**. This provides the CLK90 of the MPMC.

The above connections are shown in [Figure 3-18](#).



UG443_2_18_082007

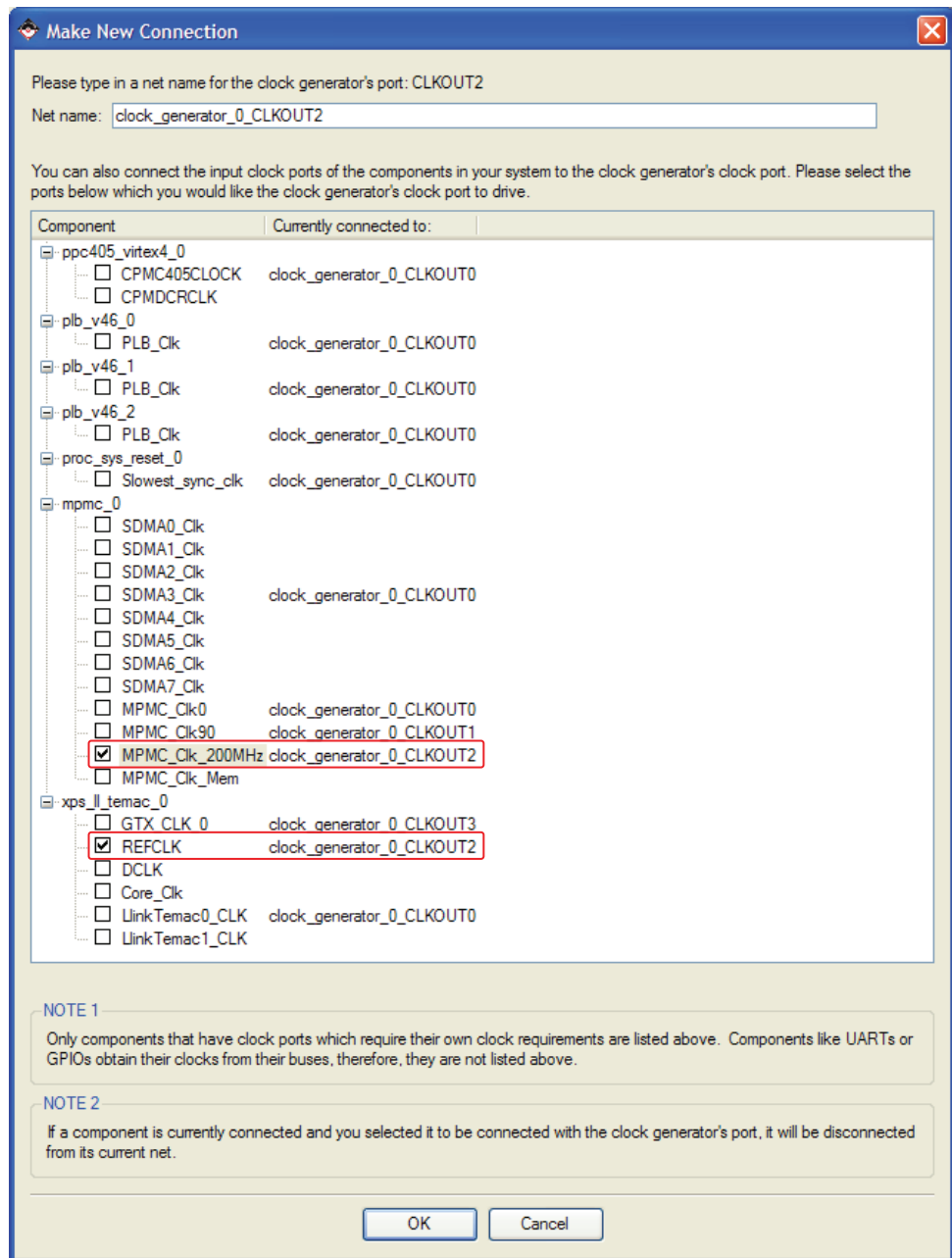
Figure 3-18: Clock Generator CLKOUT1 Connections

Under the Ports expand the Outputs tree node and click on CLKOUT2. On the right of the window, set the **Required frequency**: to 200000000 Hz, **Required phase shift** : to 0, and **Grouping Information**: to NONE. Then select **Connect to: New connection...** .

Select **MPMC_Clk_200MHz** under **mpmc_0**. This provides the 200 MHz clock for the Idelay Controllers.

Select **REF_CLK** under **xps_ll_temac_0**.

The above connections are shown in [Figure 3-19](#).



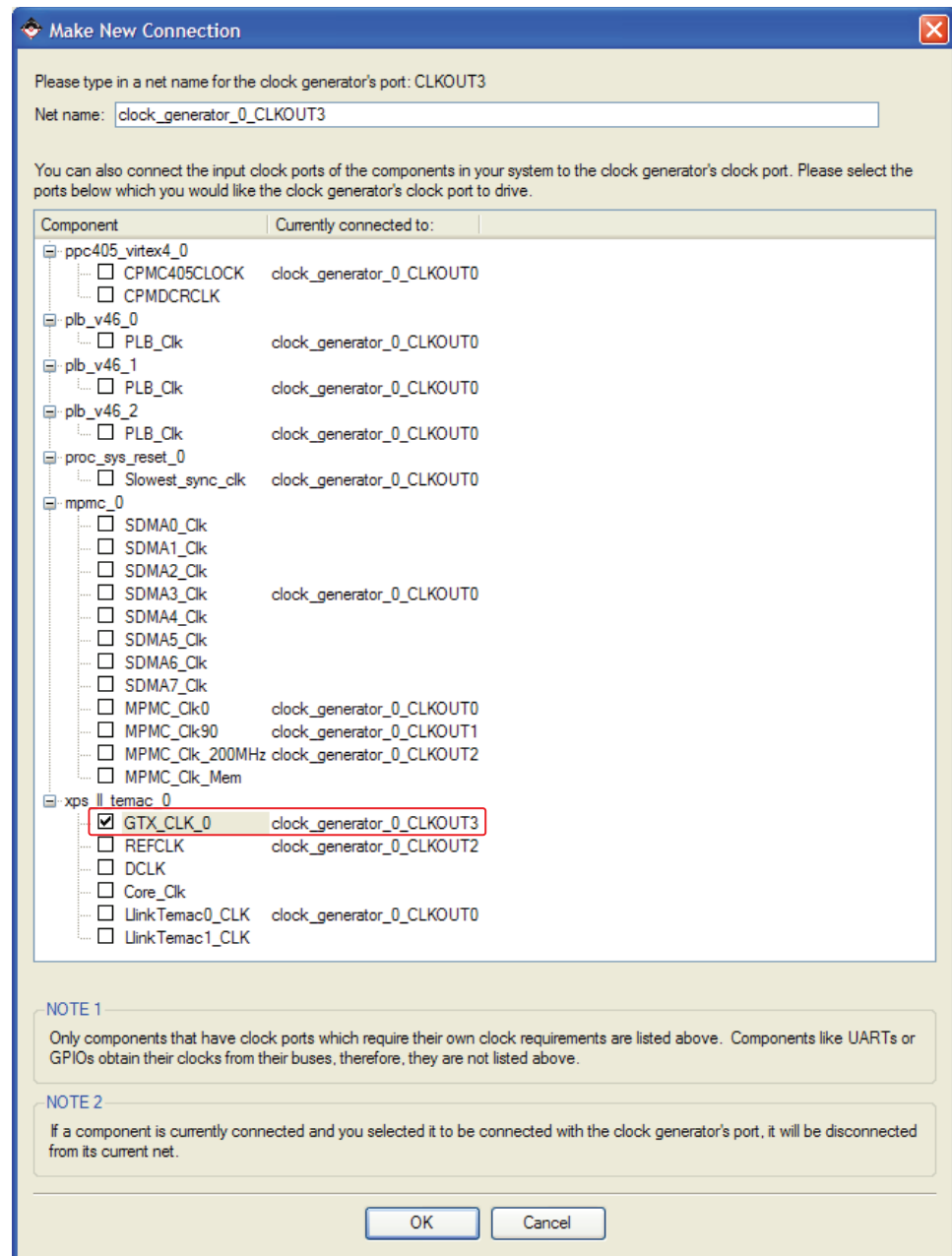
UG443_2_19_082007

Figure 3-19: Clock Generator CLKOUT2 Connections

Under the Ports expand the Outputs tree node and click on CLKOUT3. On the right of the window, set the **Required frequency:** to 125000000 Hz, **Required phase shift :** to 0, and **Grouping Information:** to NONE. Then select **Connect to: New connection...** .

Select GTX_CLK under xps_ll_temac_0.

The above connections are shown in [Figure 3-20](#).



UG443_2_20_082007

Figure 3-20: Clock Generator CLKOUT3 Connections

MicroBlaze System Migration

Introduction

This chapter describes the system migration from a MicroBlaze OPB system to a MicroBlaze PLB v4.6/XPS system. The migrated MicroBlaze PLB v4.6 system uses the Multi-Ported Memory Controller (MPMC) core for main memories like SDRAM, DDR and DDR2.

The following high-level migration steps are discussed:

1. MicroBlaze is configured for the PLB v4.6 interface
1. PLB v4.6 is added to the system where the OPB is removed
2. Clocking/Reset/Debug Scheme
3. Memory controllers are migrated to MPMC which is integrated into system
4. Ethernet Solution
5. Equivalent XPS cores (from OPB) are connected to PLB v4.6

The original and migrated systems are built for the ML505 board.

System Migration Methodology

System Hardware Migration

The method that is recommended for doing the system hardware migration is outlined below.

- Migrate the processor from MicroBlaze with an OPB interface to MicroBlaze with a PLB v4.6 interface
- Add PLB v4.6 instance to the system
- Migrate the OPB memory controller to MPMC
- Migrate the clocking/reset/debug scheme
- For peripherals, Master and Slaves cores are connected to PLB v4.6
 - ◆ The maximum number of masters that can be connected to the PLB v4.6 is 16
 - ◆ The maximum number of slaves that can be connected to the PLB v4.6 is 16

System Software Migration

No modifications to user application code are necessary when migrating from a MicroBlaze OPB system to a MicroBlaze PLB v4.6/XPS system.

Example MicroBlaze OPB System

An example MicroBlaze OPB system is shown in [Figure 4-1](#). This system shows common cores that are found inside a typical embedded system.

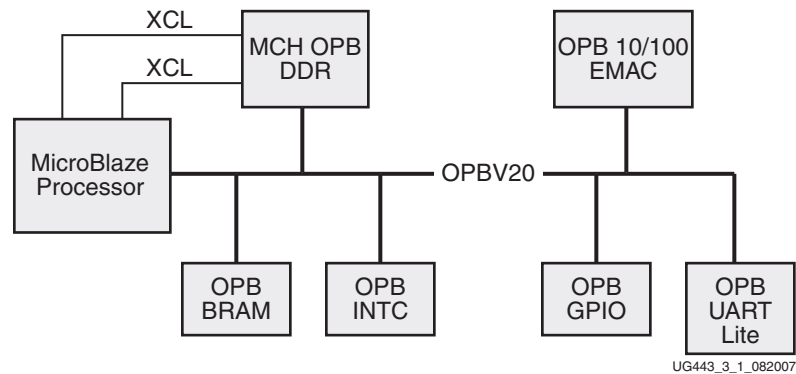


Figure 4-1: MicroBlaze OPB System

Migrated MicroBlaze PLB v4.6/XPS System

The migrated MicroBlaze PLB v4.6/XPS system is shown in [Figure 4-2](#).

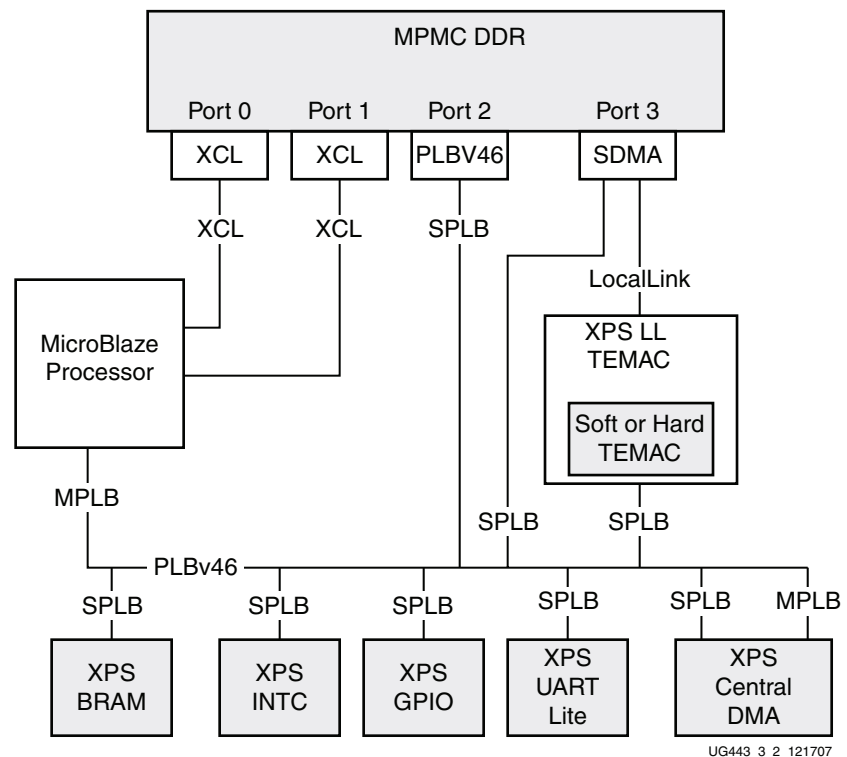


Figure 4-2: MicroBlaze PLB v4.6/XPS System

Overview of System Migration

Processor Migration

The MicroBlaze processor version is replaced with to version 7.00.a which supports a PLB v4.6 interface. Both the IXCL and DXCL bus interfaces are connected to the MPMC through XCL Port Interface Modules (PIMs). MicroBlaze provides a master PLB v4.6 connections to the PLB v4.6 shared bus to allow access to the other peripherals on the shared bus.

Interconnect

The OPB instance is deleted and the PLB v4.6 instance is added inside the system.

XPS Peripherals

This section covers the general migration from an OPB to a XPS core. This is applicable to slave and master/slave cores.

1. The former OPB core is deleted.
2. The XPS core is added to the system.
3. Parameters and ports are connected in the same manner as the OPB equivalent.
4. The slave or master/slave interfaces are connected to the PLB v4.6 instance.

In addition, the plbv46_opb_bridge and the opb_plbv46_bridge bridges are available in case an OPB core must be added to the system.

Memory Controller Migration

The memory controller for SDRAM, DDR and DDR2 memories in a PLB v4.6 system is the Multi-Ported Memory Controller (MPMC).

MPMC can support up to 8 configurable ports. Ports can be configured as XCL, PLB v4.6, SDMA (Soft Direct Memory Access), and NPI (Native Port Interface).

In the example migrated system block diagram, the MPMC is configured to support 4 ports and uses the XCL, PLB v4.6 and the SDMA Port Interface Modules (PIMs).

The XCL PIMs on MPMC are for the IXCL (Port 0) and the DXCL (Port 1) bus interfaces from the MicroBlaze processor.

The PLB v4.6 PIM (Port 2) is connected to the PLB v4.6 shared bus inside the system. This allows other masters connected to the PLB v4.6 shared bus to have access to the main memory.

Note: The PLB v4.6 PIM for the shared bus is required if a master exists on the shared bus that needs to access the MPMC address range or if MicroBlaze is not setup for XCL transactions to the MPMC.

The SDMA PIM is connected to Port3.

The SDMA PIM LocalLink interface is connected to the XPS LL TEMAC core to provide DMA transactions in and out of memory. The SDMA PIM is connected to the PLB v4.6 bus as a slave and to the XPS LL Temac through a LocalLink interface. The PLB v4.6 slave connection allows masters on the PLB v4.6 bus to setup and control DMA operations.

Ethernet Controller Migration

The migration of the OPB Ethernet controller to the XPS LL TEMAC controller varies depending on the features configured for the OPB Ethernet core.

If the OPB Ethernet core in the MicroBlaze OPB system was configured to include the Scatter-gather DMA feature, the migrated system will require a DMA Controller with Scatter-gather DMA capability to be added. The SDMA PIM for the MPMC provides Scatter-gather DMA capability to and from the MPMC. The XPS LL TEMAC connects to this DMA controller via the LocalLink port.

If the OPB Ethernet core in the MicroBlaze - OPB system was configured to use FIFO direct mode, the migrated system will require the addition of the XPS LL FIFO core. The XPS LL TEMAC connects to the XPS LL FIFO via the LocalLink port. The FIFO data is then accessed through the PLB v4.6 connection of the XPS LL FIFO.

The XPS LL TEMAC could instantiate either a hard TEMAC or a soft TEMAC. Unlike OPB Ethernet, XPS LL TEMAC will instantiate/use the Hard TEMAC for V5LXT. XPS LL TEMAC has two hard TEMACs available inside the core but only one is used for this migration example.

System Migration Inside an EDK System

Migrating Buses/Processor

Migrating the Bus

Remove the mb_opb instance (Delete instance and its internal ports) inside the System Assembly View/Bus Interfaces.

In addition, add a PLB v4.6 instance by expanding the Bus Bridge tree node. Right click on **Processor Local Bus (PLB) 4.6** and select **Add IP**. The buses are for the PLB v4.6 shared bus inside the system. The instances created is plbv46_0.

Migrating the Processor

Adding/Removing MicroBlaze Instance

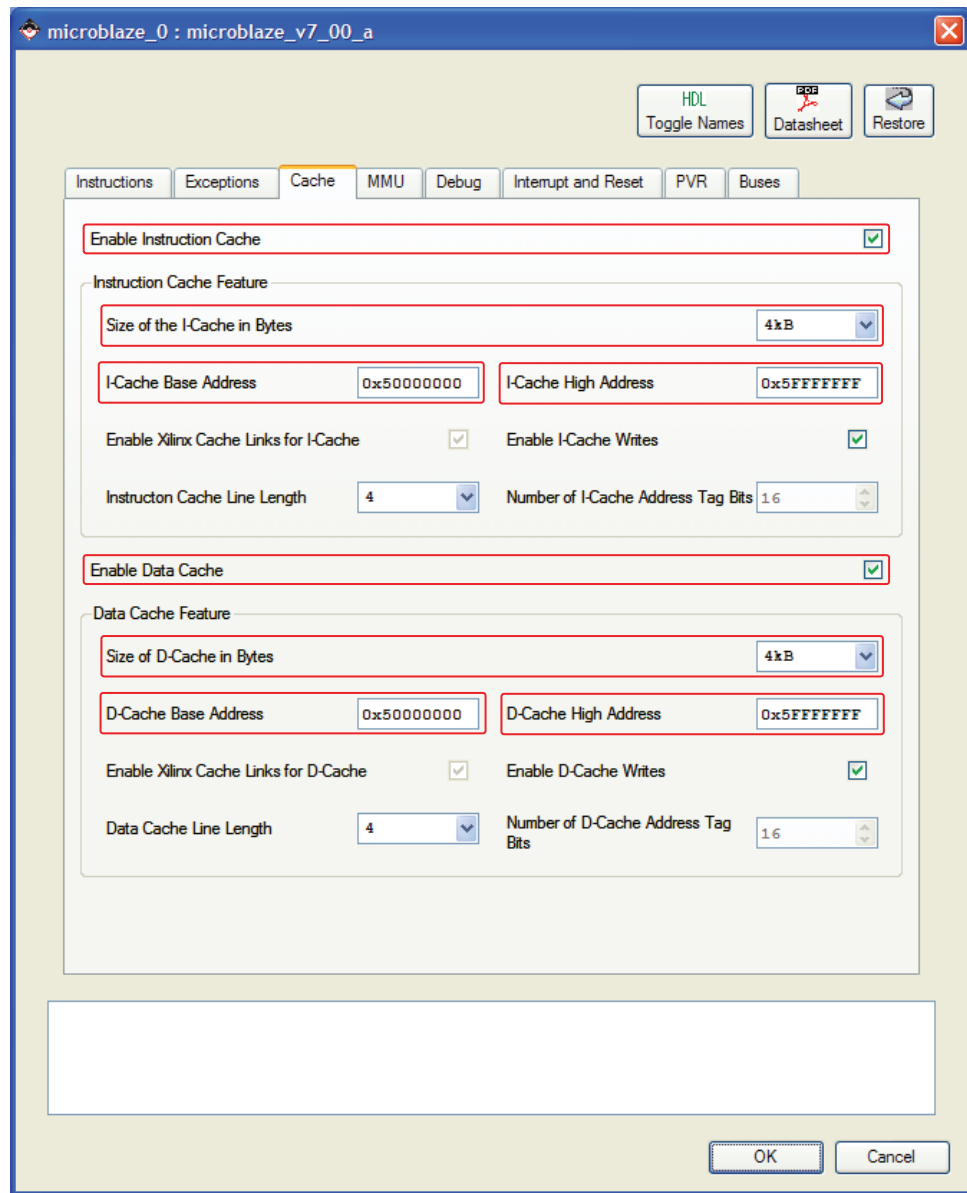
Remove the former MicroBlaze instance by right clicking on **microblaze_0** inside the **System Assembly View/Bus Interfaces** and clicking on **Delete Instance...** (Delete instance and its internal ports).

Add the latest version of the processor by expanding the Processor tree node inside the IP Catalog tab. Right click on **MicroBlaze** and click on **Add IP**. This creates the microblaze_0 instance.

Setting Parameters

Right click on microblaze_0 inside the System Assembly View and select **Configure IP ...**.

Inside the Cache Tab, **Enable Instruction Cache** is **enabled** and **Enable Data Cache** is **enabled**. **Size of the I-Cache in Bytes** and **Size of the D-Cache in Bytes** are both set to **4kB**. **I-Cache Base Address**, **D-Cache Base Address** and **I-Cache High Address**, **D-Cache High Address** are both set to **0x50000000** and **0x5FFFFFFF**. The above settings are shown in [Figure 4-3](#).

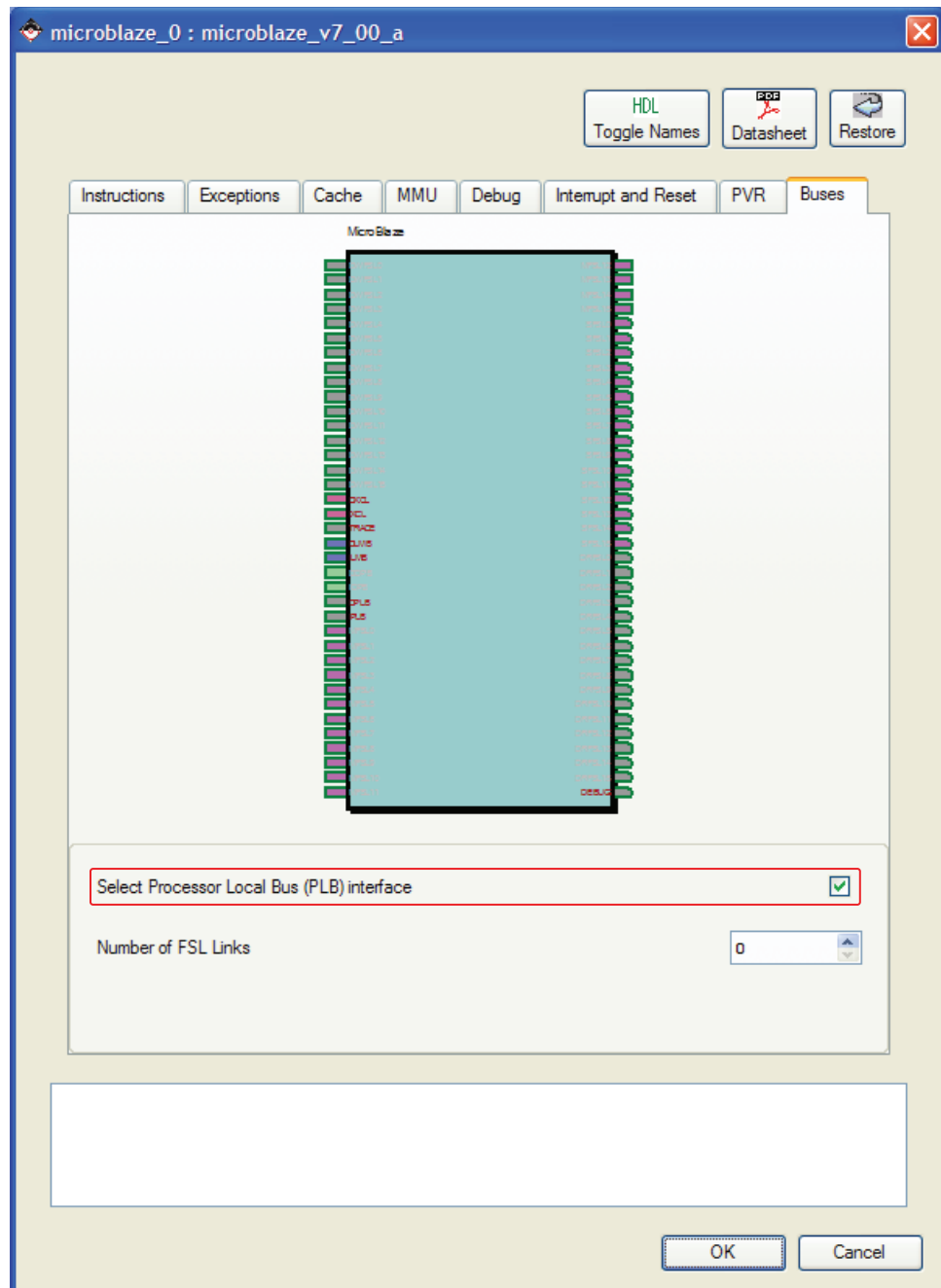


UG443_3_3_082007

Figure 4-3: Setting MicroBlaze Cache Tab

Inside the Debug Tab, select **Enable MicroBlaze Debug Module Interface**. This allows MicroBlaze to use debug. The MDM core will be discussed later in this chapter.

Inside the Buses Tab, **Select Processor Local Bus (PLB) interface** is enabled as shown in Figure 4-4.



UG443_3_4_082007

Figure 4-4: Setting MicroBlaze Buses Tab

Setting Bus Interfaces

Inside the Bus Interfaces Tab, expand the `microblaze_0` instance. Select the following bus interfaces, **IPLB** and **DPLB** bus connections are connected to `plb_v46_0`, **ILMB** is connected to `ilmb`, and **DLMB** is connected to `dlmb`. The **DEBUG** bus interface will be connected later in this chapter.

Note: LMB BRAM IF CNTLR must be version v2.10.a or higher for PLB v4.6 systems.

The above selections are shown in [Figure 4-5](#).

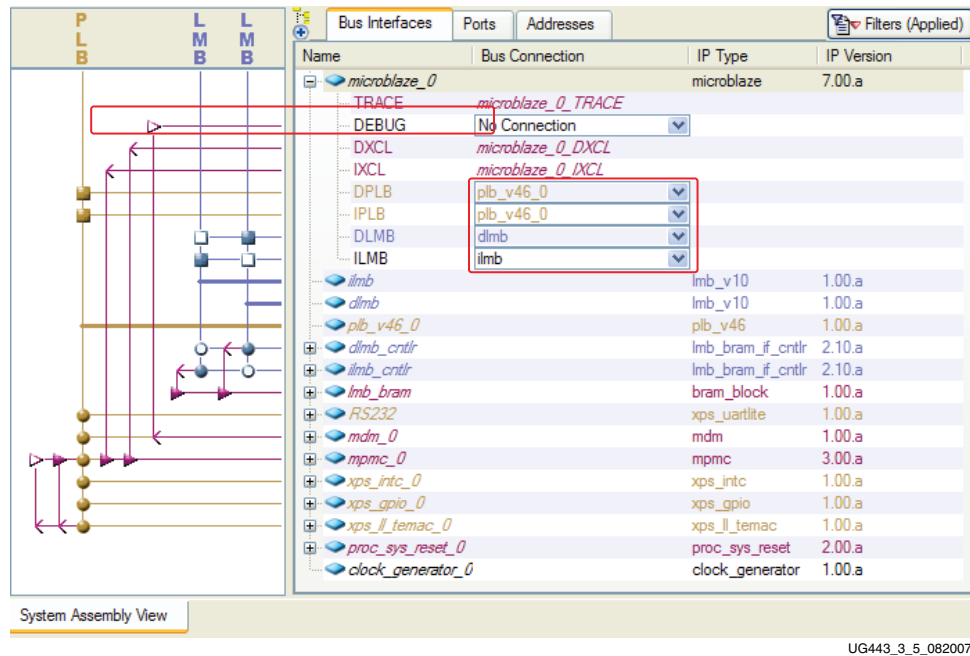


Figure 4-5: Setting MicroBlaze Bus Interfaces

Migrating The Memory Controller

Adding/Removing Memory Controllers

Remove the **DDR2_SDRAM_32Mx32** instance by right clicking on **DDR2_SDRAM_32Mx32** inside the **System Assembly View/Bus Interfaces** and clicking on **Delete Instance...** . Then click on **Delete instance and its internal ports**.

Add MPMC to the system by expanding the Memory Controller tree node inside the IP Catalog tab. Right click on **Multi-Port Memory Controller** and click on **Add IP**. This creates the `mpmc_0` instance.

The MCH OPB DDR2 memory controller provided a big endian connection to memory devices whereas MPMC provides a little endian connection to memory devices. In turn the user needs to swap the pins to match the proper endian of the MPMC. In addition, it is preferred to use a MIG complaint pinout to obtain the best performance with MPMC.

Refer to the MPMC documentation for IOSTANDARDS required for certain pins for the UCF which might be different from the previous memory controller.

Configuring The MPMC

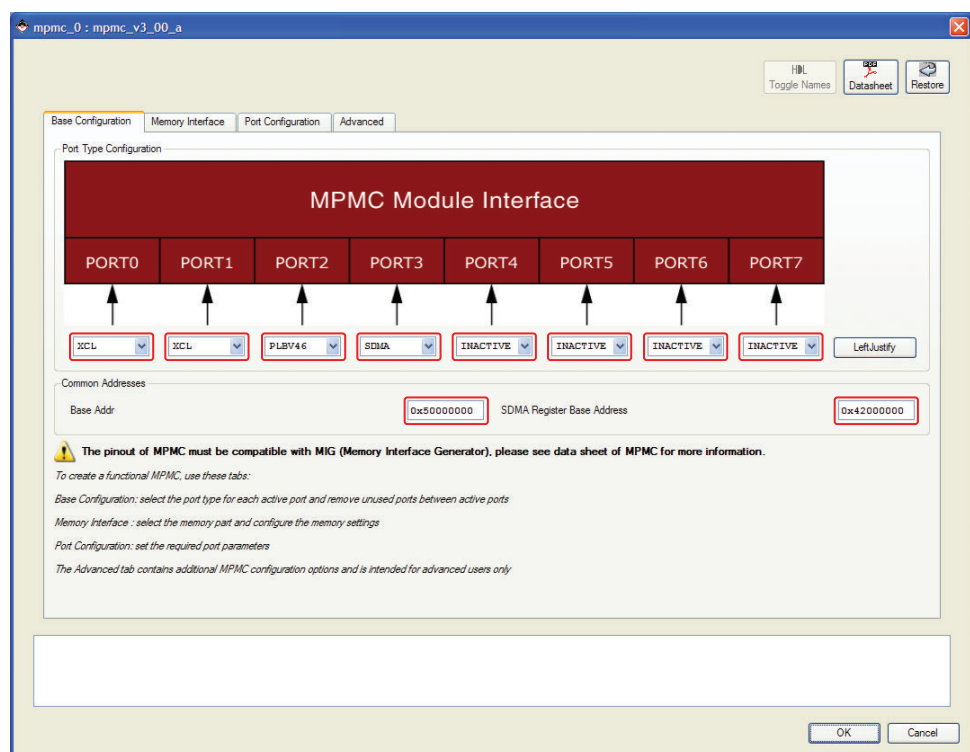
Right click on mpmc_0 inside the System Assembly View and select **Configure IP ...** .

Setting Ports

Inside the **Base Configuration** Tab, four ports will be used on the MPMC. Port 0 is set to the **Port Type** of XCL. Port 1 is set to the **Port Type** of XCL. Port 2 is set to the **Port Type** of PLBV46. Port 3 is set to the **Port Type** of SDMA.

In addition, under Common Addresses, the **Base Addr** is set to **0x50000000** which is the base address of the MCH OPB DDR2 in the existing system and the **SDMA Register Base Address** is set to **0x42000000** in this case. All ports of the MPMC are sharing the same base and high address.

The above configuration of the ports is shown in [Figure 4-6](#).



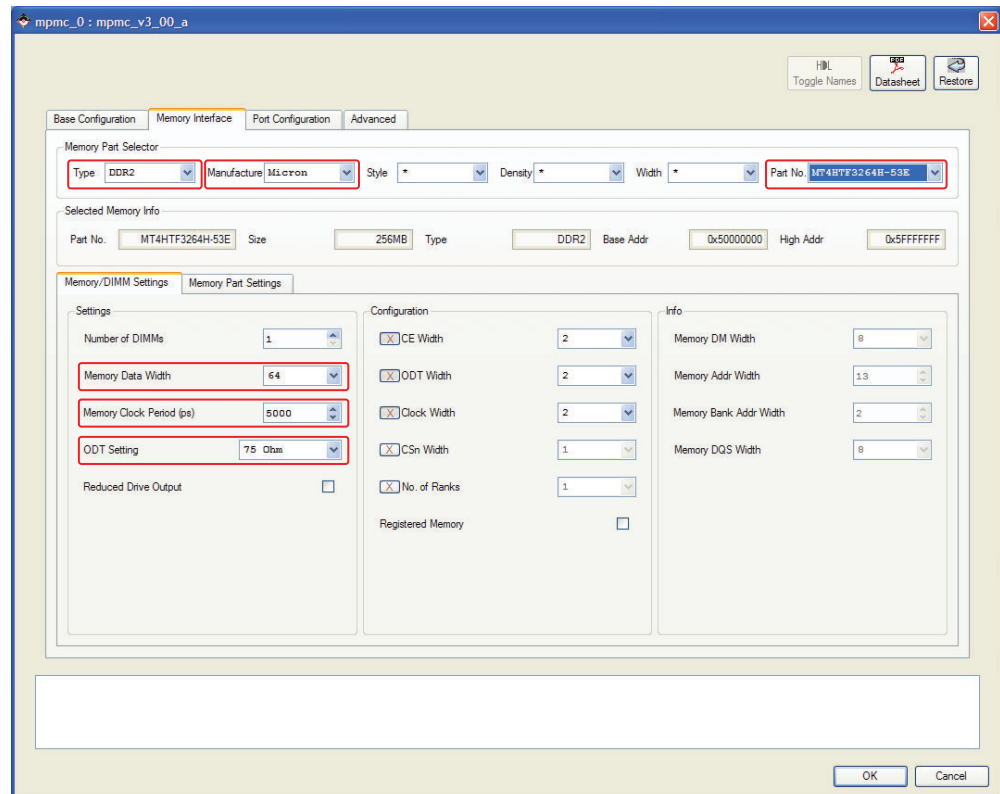
UG443_3_6_082007

Figure 4-6: MPMC Base Configuration Tab

Inside the **Memory Interface** Tab, selections are made for the DDR memory on the board. For Memory Part Filter, **Type** is set to **DDR2**, **Manufacture** is set to **Micron**, and **Select a Part** is set to **MT4HTF3264H-53E**. The Part Settings, Memory Timing Information and DIMM Settings sections are set automatically based upon the DDR2 selected inside the Memory Part Selector section.

For the Memory System Settings section, **Memory Clock Period** is set to **5000ps**, **Memory Data Width** is set to **64**, and **ODT Setting** is set to **75 Ohm**. In addition, set **ODT Width** to **2** and **Clock Width** to **2**. These settings are for a 64-bit DDR2 data width.

The above selections are shown [Figure 4-7](#).



UG443_3_7_082007

Figure 4-7: MPMC Memory Interface Tab

Inside the Port Configuration Tab, Port 0 - XCL, **Write Transfer** is set to 0. This port is used for the instruction side of XCL on the MicroBlaze processor, where only memory read transactions occur.

In addition, Port 3 - SDMA, set the **MPMC to SDMA Clk Ratio** to 2.

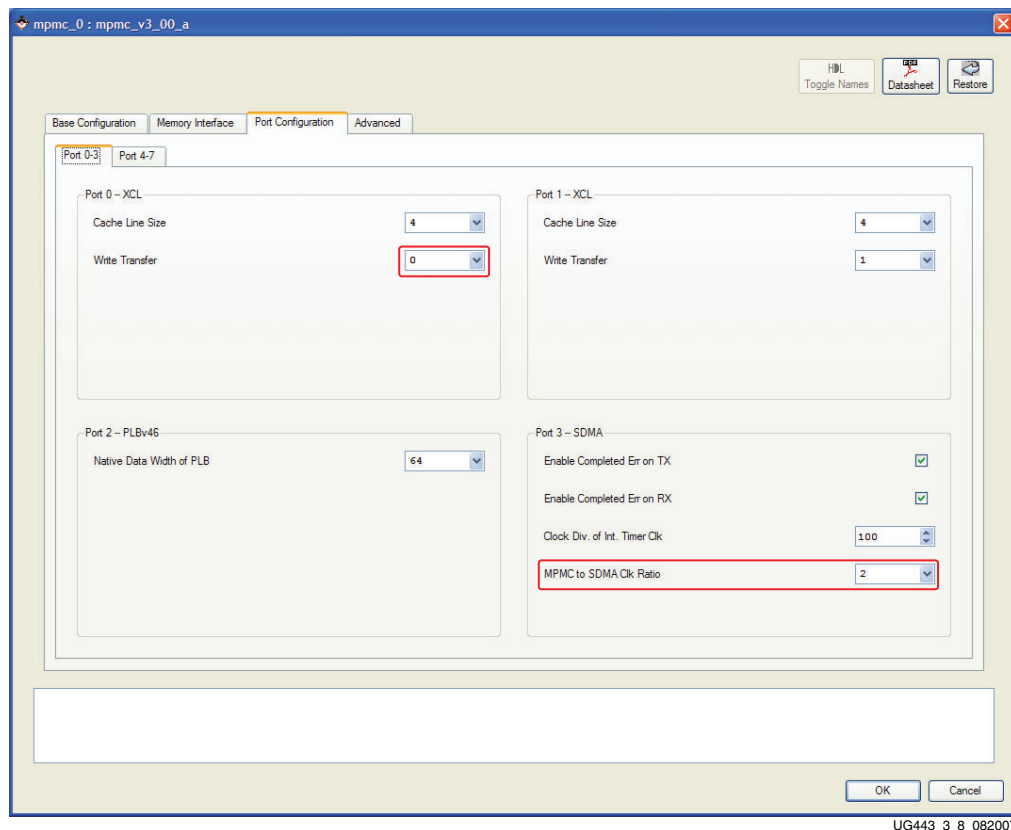


Figure 4-8: MPMC Port Configuration Tab

Inside the Advanced/Misc Tab, set the **C_NUM_IDELAYCTRL** to 3 and **C_IDELAYCTRL_LOC** to **IDELAYCTRL_X0Y5-IDELAYCTRL_X0Y1-IDELAYCTRL_X0Y0**. These settings are for the ML505 board.

MPMC Bus Interfaces

Inside the Bus Interfaces Tab, expand the mpmc_0 instance.

XCL0 and XCL1 are connected to **microblaze_0_IXCL** and **microblaze_0_DXCL** bus interfaces on the MicroBlaze processor.

SPLB2 is connected to **plb_v46_0**.

SDMA_CTRL3 is connected to **plb_v46_0** which is the slave interface of the SDMA. The LocalLink interface is connected later in this chapter for the XPS LL Temac.

The connections are shown in [Figure 4-9](#).

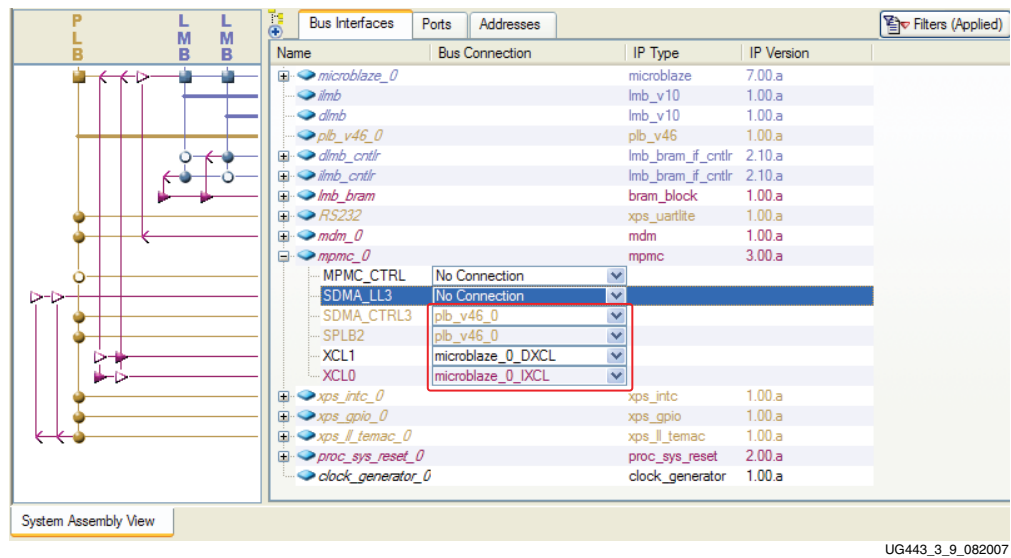


Figure 4-9: MPMC PLB v4.6 PIM Bus Interfaces

MPMC Ports

The MPMC DDR2 ports are connected similar to the MCH OPB DDR2 ports. The external ports for DQ and DQS external ports the Name and Net name must be the same. Discussion of connecting the system clocking needed for the memory controller is discussed later in this chapter.

In addition, the SDMA TX and RX interrupts should be connected to interrupt controller inside the system.

Migrating Ethernet Solution

Adding/Removing Ethernet

Remove the Ethernet_MAC instance by right clicking on **Ethernet_MAC** inside the **System Assembly View/Bus Interfaces** and clicking on **Delete Instance...** . Then click on **Delete instance and its internal ports.**

Add XPS LL TEMAC to the system by expanding the Communication High-Speed tree node inside the IP Catalog tab. Right click on **XPS LocalLink Tri-mode Ethernet Mac** and click on **Add IP.** This creates the xps_ll_temac_0 instance.

Setting Parameters

Right click on **xps_ll_temac_0** inside the System Assembly View and select **Configure IP ...**

The XPS LL TEMAC is configured to use a single Hard TEMAC by unselecting **Enable TEMAC1.** The GMII interface is used by setting **Physical Interface Type** to **GMII.** The existing system used a MII interface since the OPB Ethernet didn't allow for the Hard TEMAC to be instantiated. **Ratio of PLB Bus Clock to Core Clock** is set to **1.**The above selections are shown in [Figure 4-10.](#)

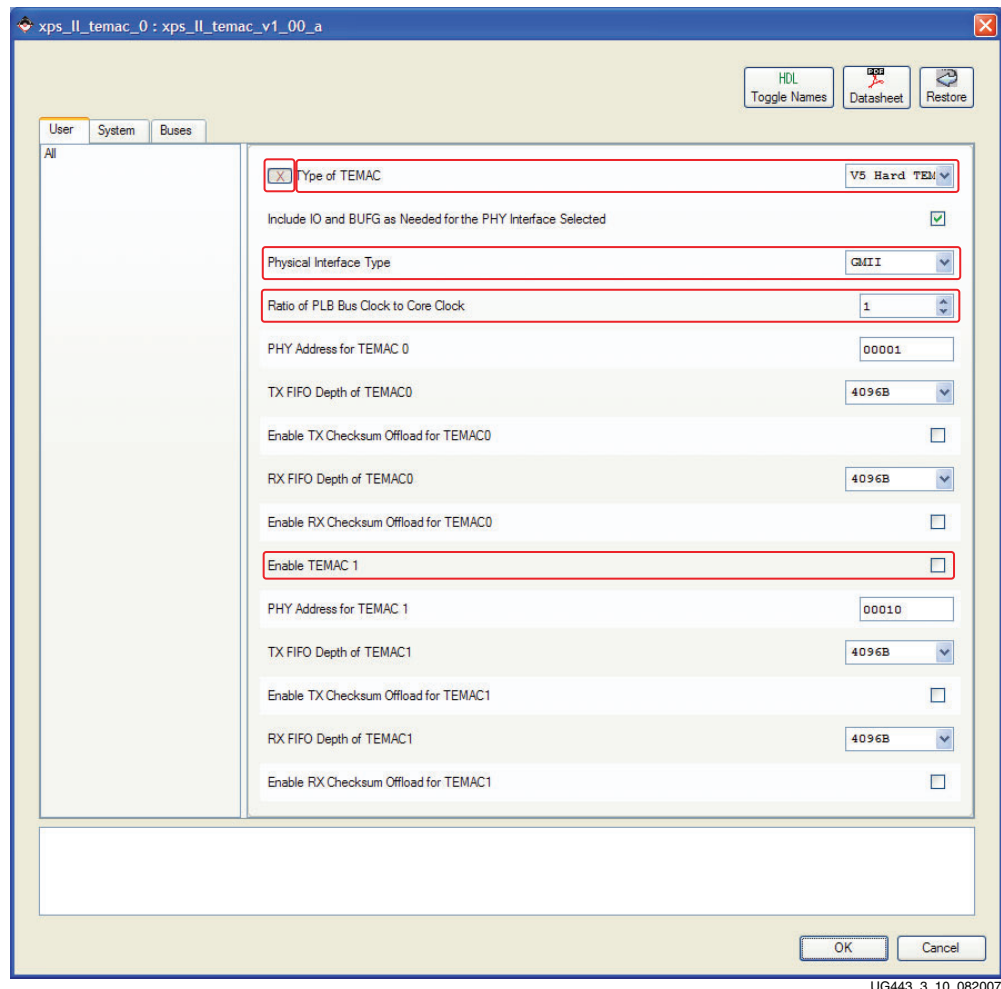
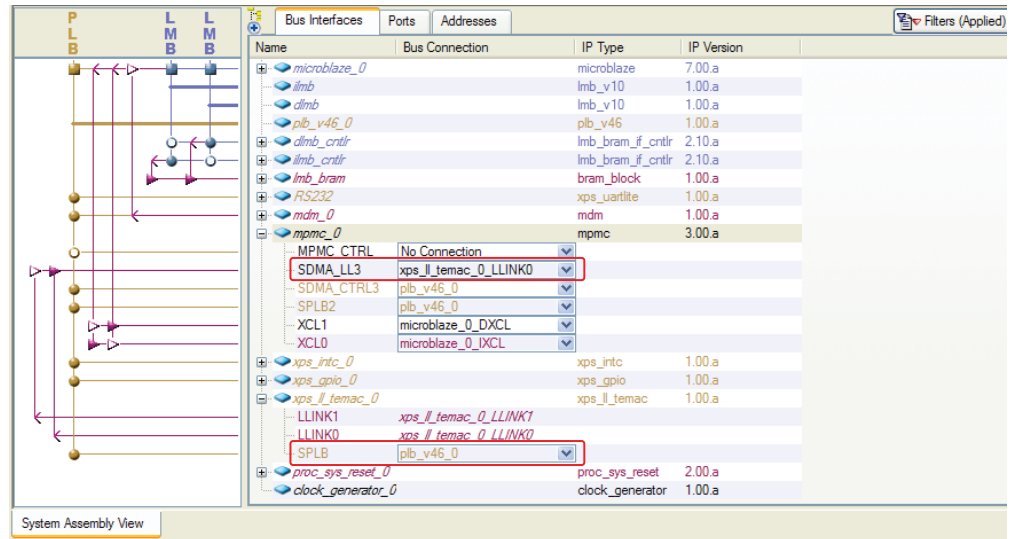


Figure 4-10: XPS LL TEMAC Parameters

Setting Bus Interfaces

The slave interface of the XPS LL TEMAC core is connected to the plb_v46_0 instance inside the system, as shown in Figure 4-10.

Port 3 of the MPMC contains the SDMA. The LocalLink bus interface from the XPS LL TEMAC is connected to the SDMA LocalLink bus interface on the MPMC, as shown in Figure 4-11.



UG443_3_11_082007

Figure 4-11: XPS LL TEMAC Bus and LocalLink Interfaces

XPS LL TEMAC Ports

The XPS LL TEMAC GMII ports are connected for GMII operation. Discussion of connecting the system clocking for the XPS LL TEMAC is discussed later in this chapter.

Migrating XPS Peripherals

In the migration example, the OPB INTC, OPB GPIO, and OPB UART Lite cores are replaced with the equivalent XPS INTC, XPS GPIO, and XPS UART Lite cores respectively.

Expand the tree node for **microblaze_0**. The Interrupt output port of the Interrupt Controller (IRQ) should be connected to the input Interrupt port on MicroBlaze.

Adding these cores are not discussed.

Modifying Clocking/Reset/Debug Inside The System

The processor system reset module is added inside the system. Also, the debug module is added to the system. In addition, the DCMs instantiations inside the system are replaced with the clock generator core.

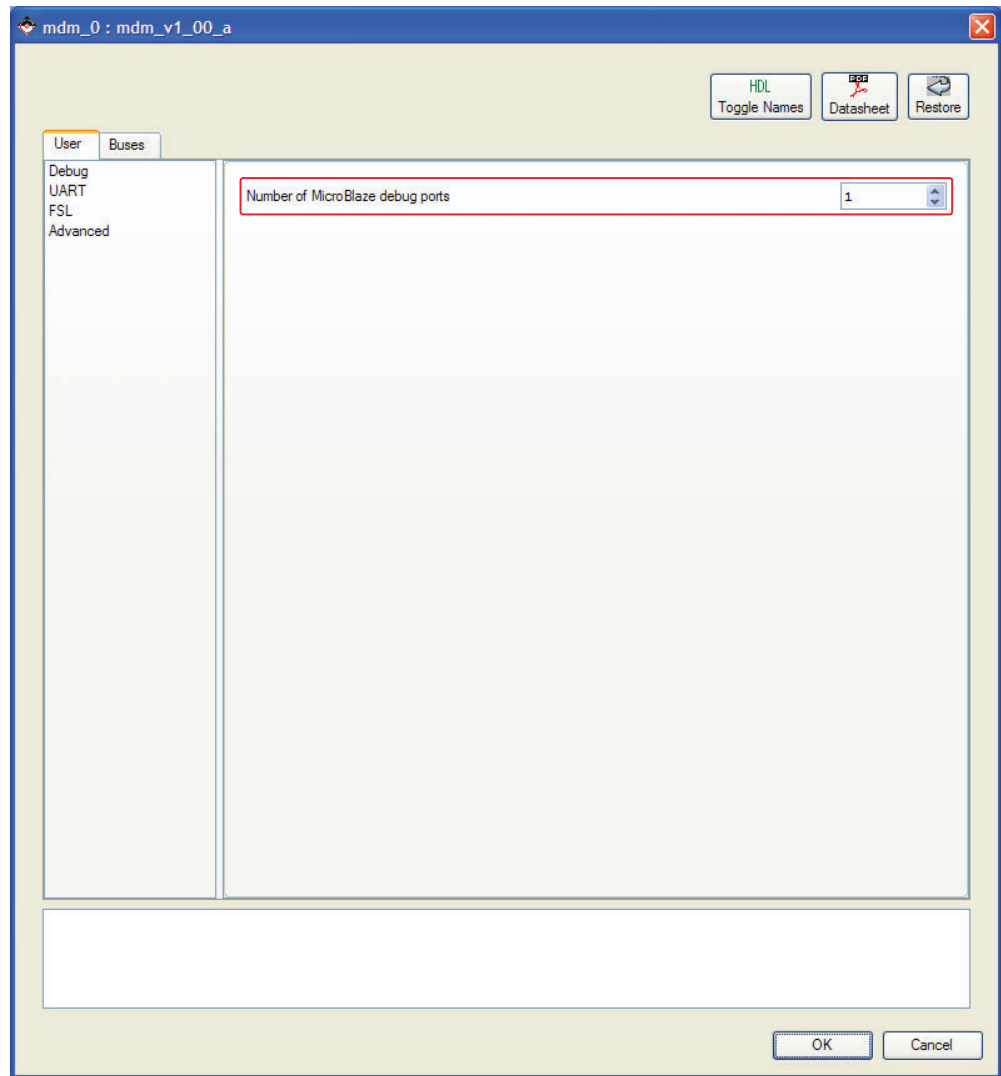
Debug Module

Add the **MDM** to the system by expanding the Debug tree node inside the **IP Catalog** tab. Right click on **MicroBlaze Debug Module (MDM)** and click on **Add IP**. This creates the `mdm_0` instance.

For MicroBlaze systems, the MDM core is used inside the system for debug.

Since the MDM core can be used on either PLB v4.6 or OPB systems, the parameters, bus connections, and ports are set to PLB v4.6.

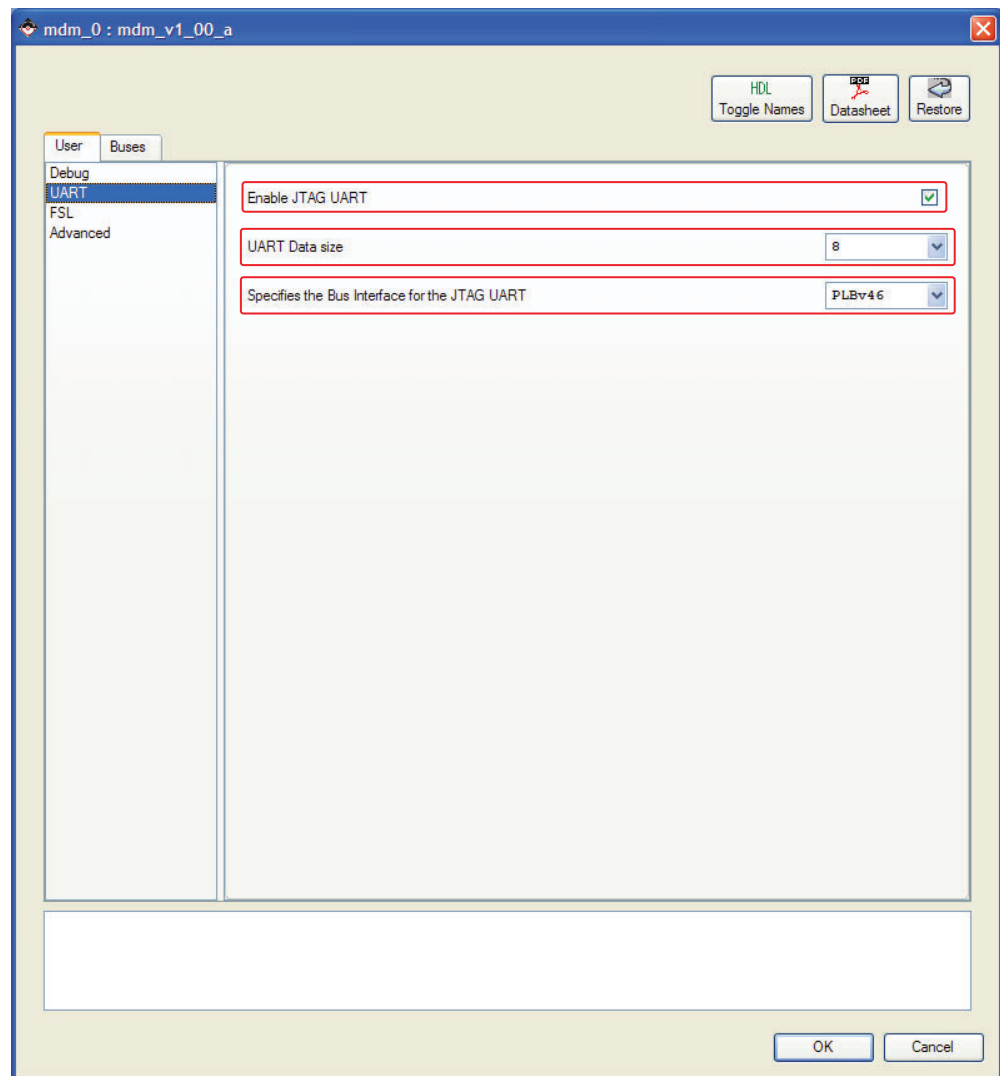
Inside the **User/Debug** Tab, **C_MB_DBG_PORTS** is set to **1** since only one processor is used inside the system. The above parameter is shown in [Figure 4-12](#).



UG443_3_12_082007

Figure 4-12: MDM Parameters User/Debug

Inside the **User/UART** Tab, the MDM parameter **Specifies the Bus Interface for the JTAG UART** is set to **PLBv46**. **Enable JTAG UART** is selected and **UART Data Size** is set to **8** as shown in [Figure 4-13](#).



UG443_3_13_082007

Figure 4-13: MDM Parameters User/UART

The individual debug signals for MicroBlaze are grouped to the DEBUG bus interface. The MDM core has separate debug bus interfaces for each debug port. The **DEBUG** bus interface on MicroBlaze is connected to **mdm_0_MBDEBUG_0** on the MDM core as shown in Figure 4-14. In addition, the **SPLB** for the MDM core is connected to the **plb_v46_0** instance inside the system.

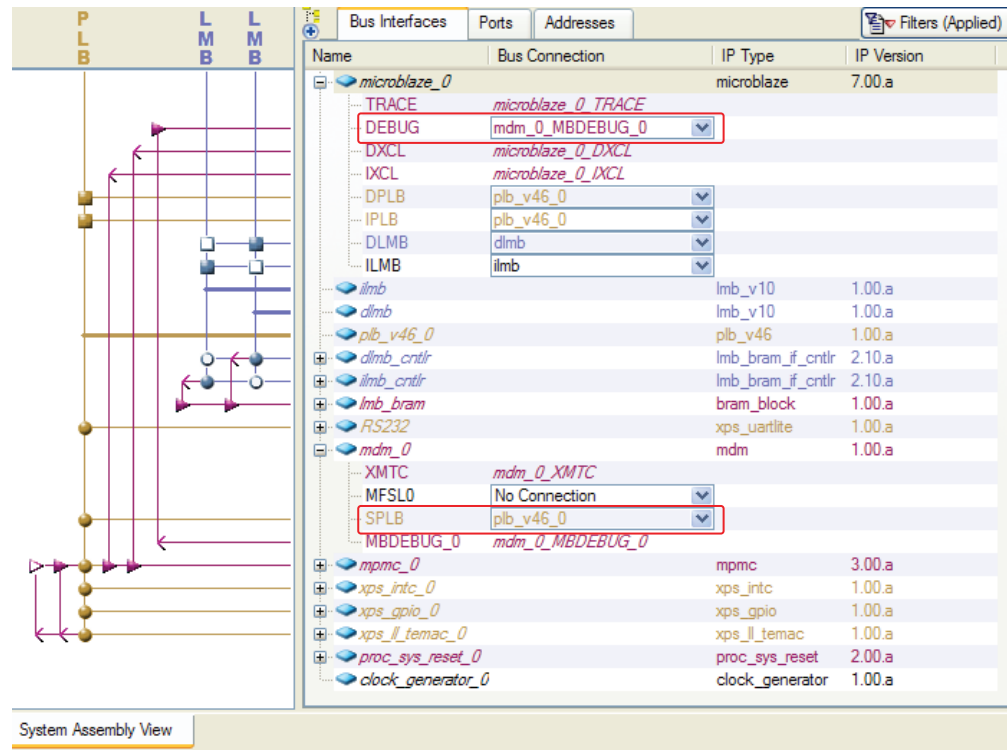


Figure 4-14: MDM Bus Interfaces

Since PLB v4.6 doesn't have a debug reset input like OPB, the MDM core has the Debug_SYS_Rst output port for system resets.

Expand the tree node for **mdm_0** inside the **System Assembly View/Port**. Click on **New Connection** on the Net name for **Debug_Sys_Rst**. This port is connected to the **Processor System Reset Module** which is discussed inside the next section.

Processor System Reset

The Processor System Reset Module provides resets to the MicroBlaze system. The same reset scheme is used inside Base System Builder (BSB) for MicroBlaze PLB v4.6 systems.

In a MicroBlaze system, the Processor System Reset Module provides resets to the processor, buses, and peripherals. The inputs to the module is the external system reset, slowest synchronous clock, and the DCM Lock signal of the last DCM inside the system. Refer to the Processor System Reset Module Product Specification for more information.

Inside a MicroBlaze system, the **MB_Reset** of the MicroBlaze processor is connected to the **MB_Reset** output on the Processor System Reset Module.

In addition, the **Debug_SYS_Rst** output from the MDM core is connected to the **MB_Debug_Rst** on the Processor System Reset Module. This connection allows the processor and the buses to reset through the MDM by means of the Processor System Reset Module.

The **Bus_Struct_Reset** port is connected to the bus resets inside the system.

For master and slave peripherals connected to the PLB v4.6, the peripherals get vectorized resets for each master and slave connection from the bus core.

A block diagram of the reset scheme for the MicroBlaze processor, Processor System Reset Module and the MDM pcore is shown in [Figure 4-15](#).

Note: The slowest synchronous clock, DCM Lock and bus resets connections inside the system are not shown inside the block diagram.

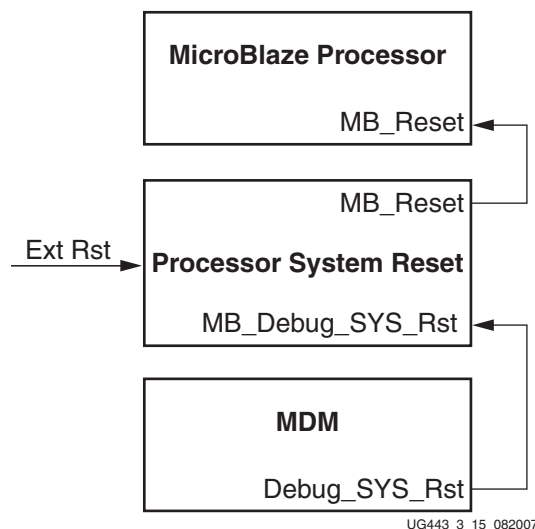


Figure 4-15: Reset Scheme Block Diagram

Adding Processor System Reset to Migrated System

Add the **Processor System Reset** by expanding the Clock Reset and Interrupt tree node inside the **IP Catalog** tab. Right click on **Processor System Reset Module** and click on **Add IP**. This creates the `proc_sys_reset_0` instance.

Right click on `proc_sys_reset_0` inside the **System Assembly View** and select **Configure IP ...**.

Set **External Active High** to 0, Number of **Bus Structure Reset Registered Outputs** set to 2. The above parameters are shown in [Figure 4-16](#).

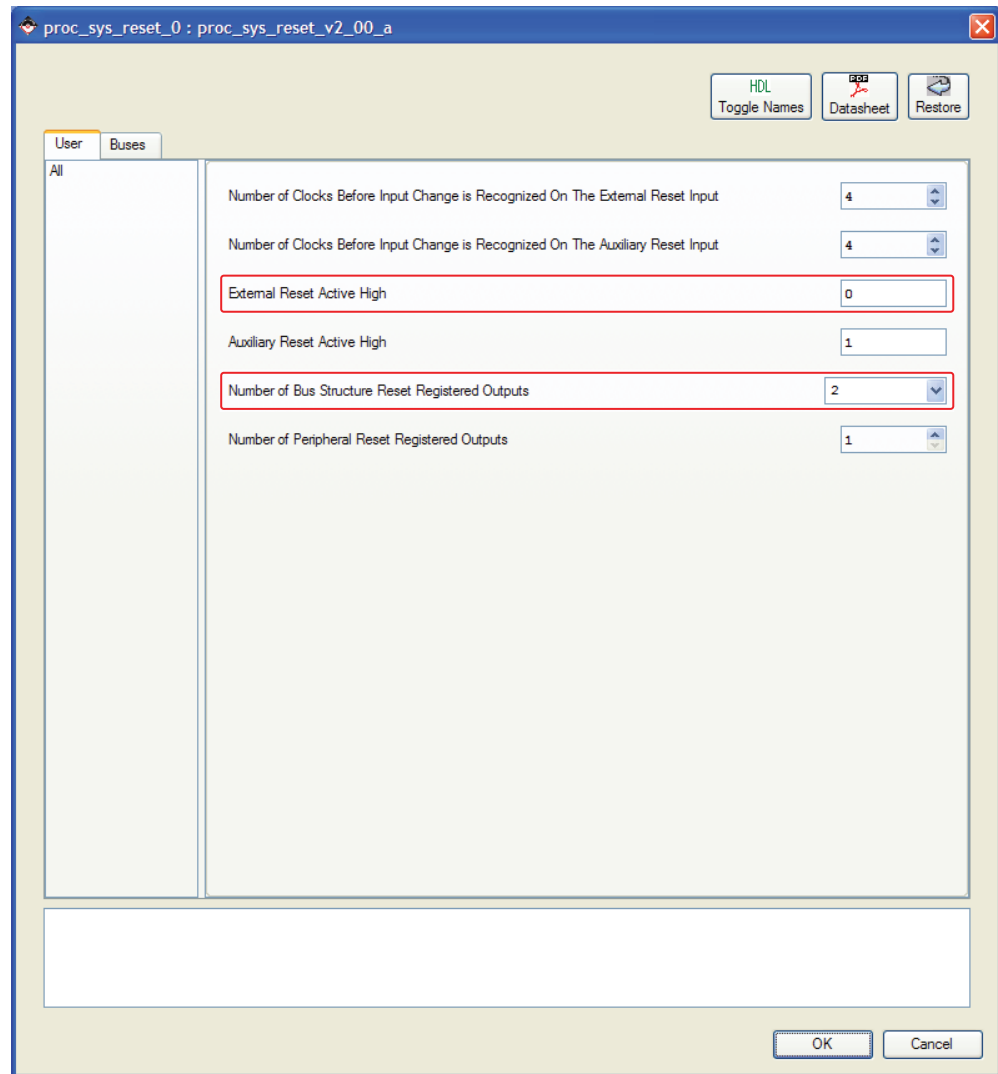
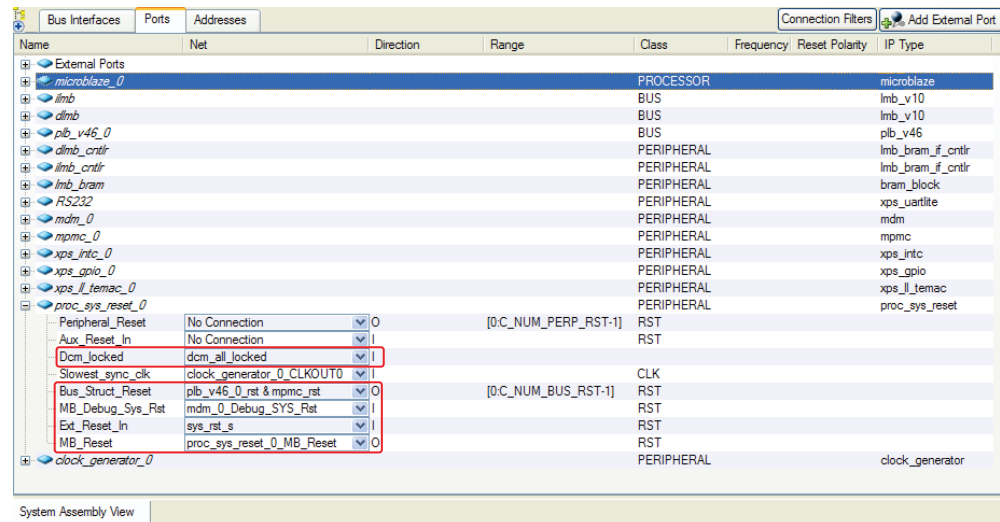


Figure 4-16: Processor System Reset Parameters

Inside the **Ports** Tab, expand the tree node for **proc_sys_reset_0**. **Bus_Struct_Reset** to **plbv46_0_rst**, and **mpmc_rst**. For **MB_Reset** set the Net name to **New Connection**. In addition, set **Dcm_locked** to **dcm_all_locked** which the signal is created inside the clocking section. **MB_Debug_Sys_Rst** is connected to **mdm_0_Debug_SYS_Rst** which is an out from the MDM core. Set **Ext_Reset_in** to **sys_rst_s**.

The following port connections are shown in [Figure 4-17](#).



Name	Net	Direction	Range	Class	Frequency	Reset Polarity	IP Type
External Ports							
microblaze_0				PROCESSOR			microblaze
lmb				BUS			lmb_v10
dlmb				BUS			lmb_v10
plb_v46_0				BUS			plb_v46
lmb_cntlr				PERIPHERAL			lmb_bram_if_cntlr
lmb_cntlr				PERIPHERAL			lmb_bram_if_cntlr
lmb_bram				PERIPHERAL			bram_block
RS232				PERIPHERAL			xps_uartlite
mdm_0				PERIPHERAL			mdm
mpmc_0				PERIPHERAL			mpmc
xps_intc_0				PERIPHERAL			xps_intc
xps_gpio_0				PERIPHERAL			xps_gpio
xps_ll_temac_0				PERIPHERAL			xps_ll_temac
proc_sys_reset_0				PERIPHERAL			proc_sys_reset
Peripheral_Reset	No Connection	O	[0:C_NUM_PERP_RST-1]	RST			
Aux_Reset_In	No Connection	I		RST			
Dcm_locked	dcm_all_locked	I					
Slowest_sync_clk	clock_generator_0_CLKOUT0	I		CLK			
Bus_Struct_Reset	plb_v46_0_rst & mpmc_rst	O	[0:C_NUM_BUS_RST-1]	RST			
MB_Debug_Sys_Rst	mdm_0_Debug_SYS_Rst	I		RST			
Ext_Reset_In	sys_rst_s	I		RST			
MB_Reset	proc_sys_reset_0_MB_Reset	O		RST			
clock_generator_0				PERIPHERAL			clock_generator

UG443_3_17_082007

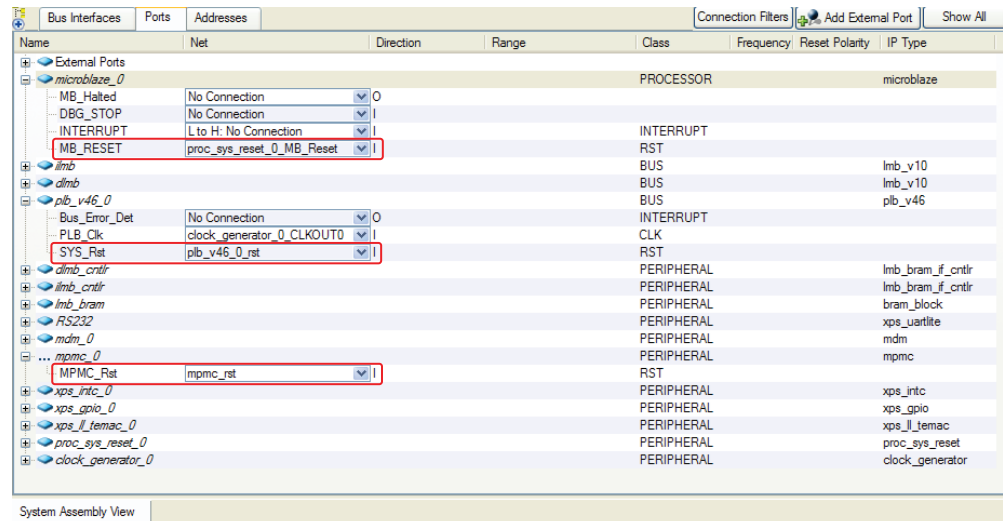
Figure 4-17: Processor System Reset Port Connections

Expand the tree node for **microblaze_0**, inside the **System Assembly View/Port**. Connect the **MB_RESET** to **proc_sys_reset_0_MB_Reset**.

Expand the tree node for **plb_v46_0**, inside the **System Assembly View/Port**. Connect the **SYS_Rst** to **plbv46_0_rst**.

Expand the tree node for **mpmc_0** and set the **MPMC_Rst** port to **mpmc_rst**.

The above connections are shown in **Figure 4-18**.



UG443_3_18_082007

Figure 4-18: Processor System Reset System Connections

Clock Generator

Overview of Existing Clocking Scheme

Originally, the system contains 2 cascading DCMs. The first DCM provides the 100 MHz clocks for the bus, and processor clock. In addition, the first DCM provides the 200 MHz clock for the next DCM that provides the DDR2 clocking. In addition, the first DCM provides the 25 MHz for the Cal_Clk for the DDR2 memory controller.

The second DCM provides the 200 MHz CLK0 and CLK90 for the MCH OPB DDR2 memory controller.

With the MCH OPB DDR2 clocks, inverters are added by BSB inside the system which are deleted inside the migrated system.

Clocking Requirements of Migrated System

The migrated system requires a 100 MHz clock for the processor and bus. MPMC requires a 200 MHz clock for the IDELAY Controllers and a 200 MHz CLK0 and CLK90 clocks. For a GMII interface for XPS LL Temac, a 125 MHz clock is needed.

Configuring Clock Generator Inside The Migrated System

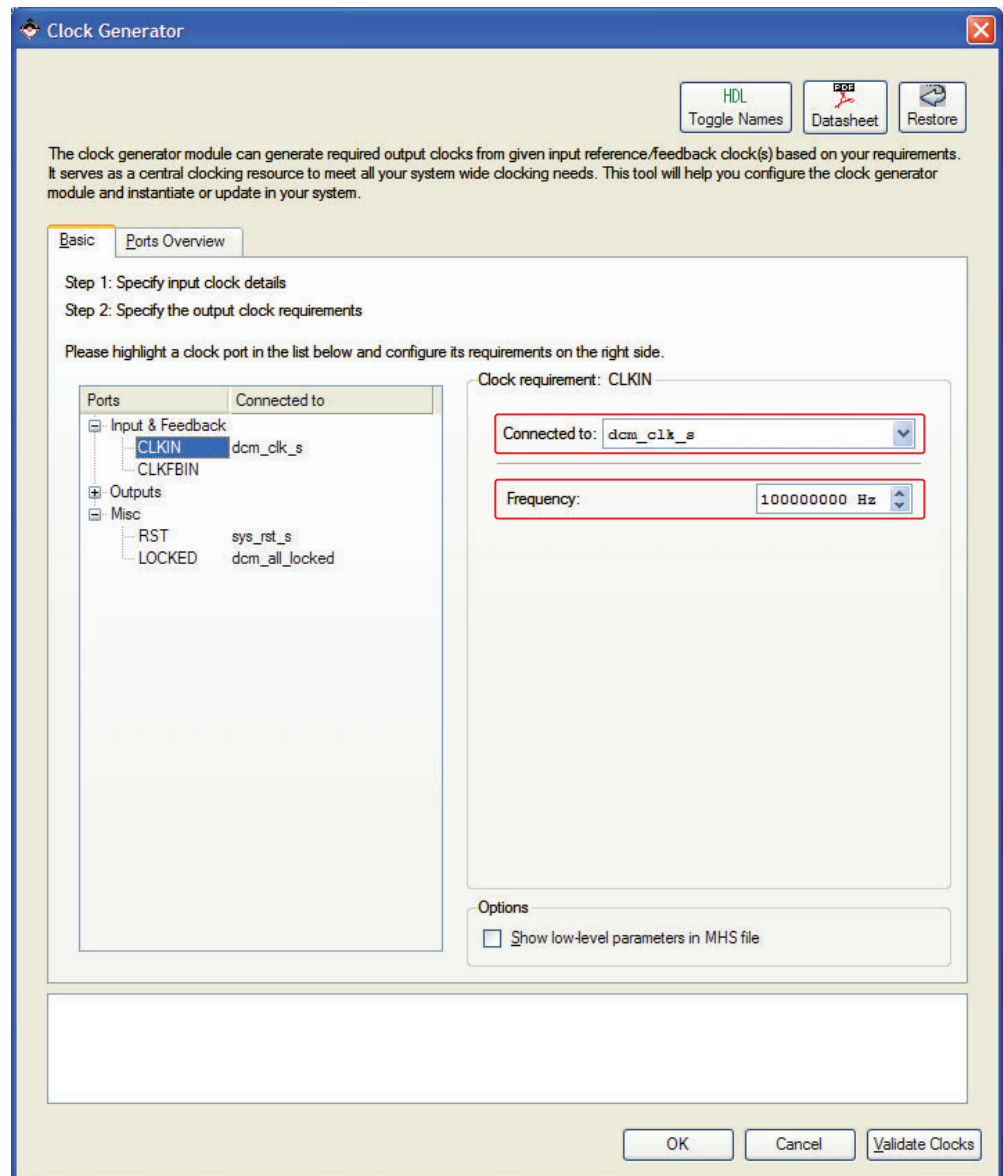
Inside the **System Assembly View/Bus Interfaces**, delete the **dcm_0** and **dcm_1** instances (Delete instance and its internal ports).

Add the clock generator by expanding the **Clock Reset and Interrupt** tree node inside the **IP Catalog** tab. Right click on **Clock Generator** and click on **Add IP**. This creates the **clock_generator_0** instance inside the system.

Right click on **clock_generator_0** inside the **System Assembly View** and select **Configure IP ...**.

Inside the **Basic** tab, this is where the user sets up the inputs and outputs to the Clock Generator.

Under the **Ports**, expand the **Input & Feedback** tree node. Click on **CLKIN**. On the right side of the window, select **Connect to: dcm_clk_s**. In addition, change the **Frequency:** to **100000000 Hz** as shown in Figure 4-19.



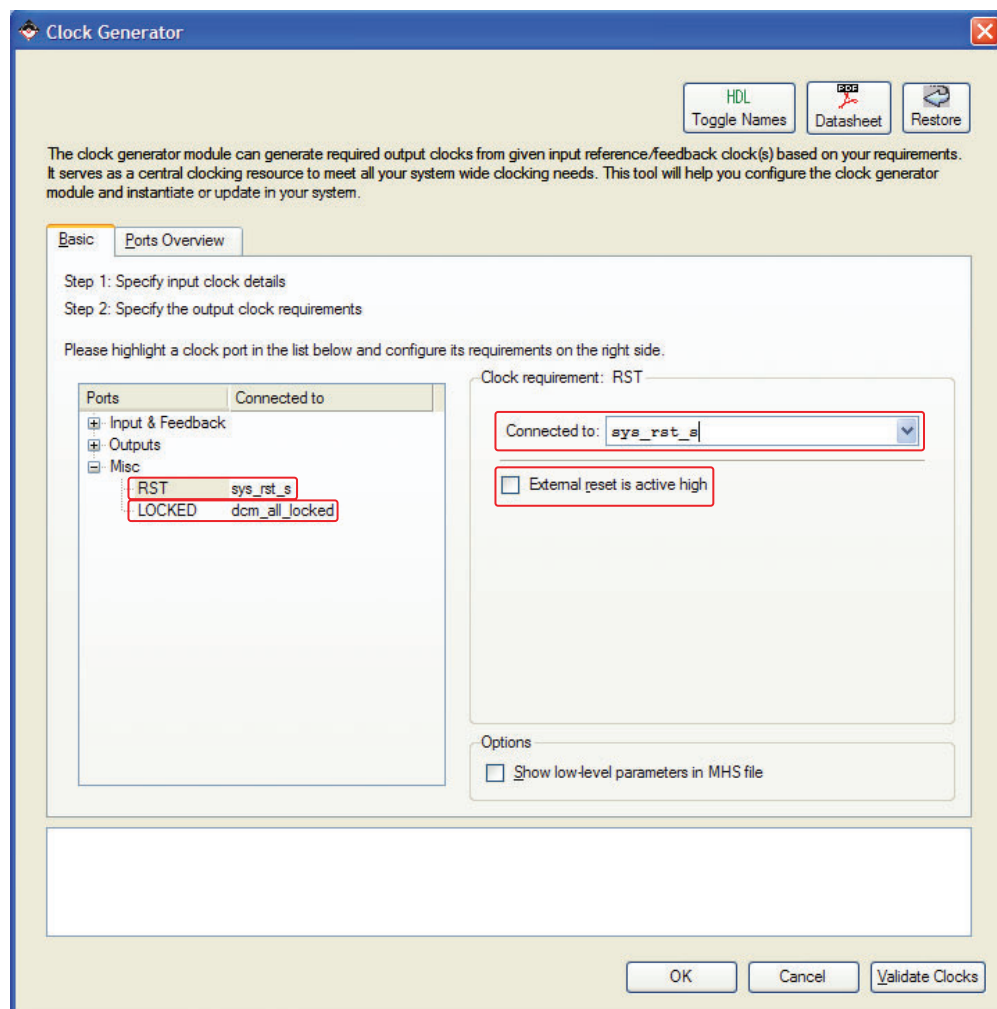
UG443_3_19_082007

Figure 4-19: Clock Generator Input and Feedback

Under the **Ports**, expand the **Misc** tree node which is near the bottom. Click on **RST**. On the right side of the window, select **Connect to: sys_rst_s**. Do not select **External reset is active high** since the polarity on the reset is active low on the board.

Click on **LOCKED**. On the right side of the window, select **Connect to: dcm_all_locked**. This output is connect to an input on the processor system reset.

The above connections are shown in [Figure 4-20](#).



UG443_3_20_082007

Figure 4-20: Clock Generator Miscellaneous

Under the **Ports** expand the **Outputs** tree node and click on **CLKOUT0**. On the right of the window, set the **Required frequency:** to **100000000 Hz**, **Required phase shift :** to **0**, and **Grouping Information:** to **None**. Then select **Connect to: New connection...**

This brings up a dialog box and select **Slowest_sync_clk** under **proc_sys_reset_0**.

Select **PLB_Clk** under **plb_v46_0**.

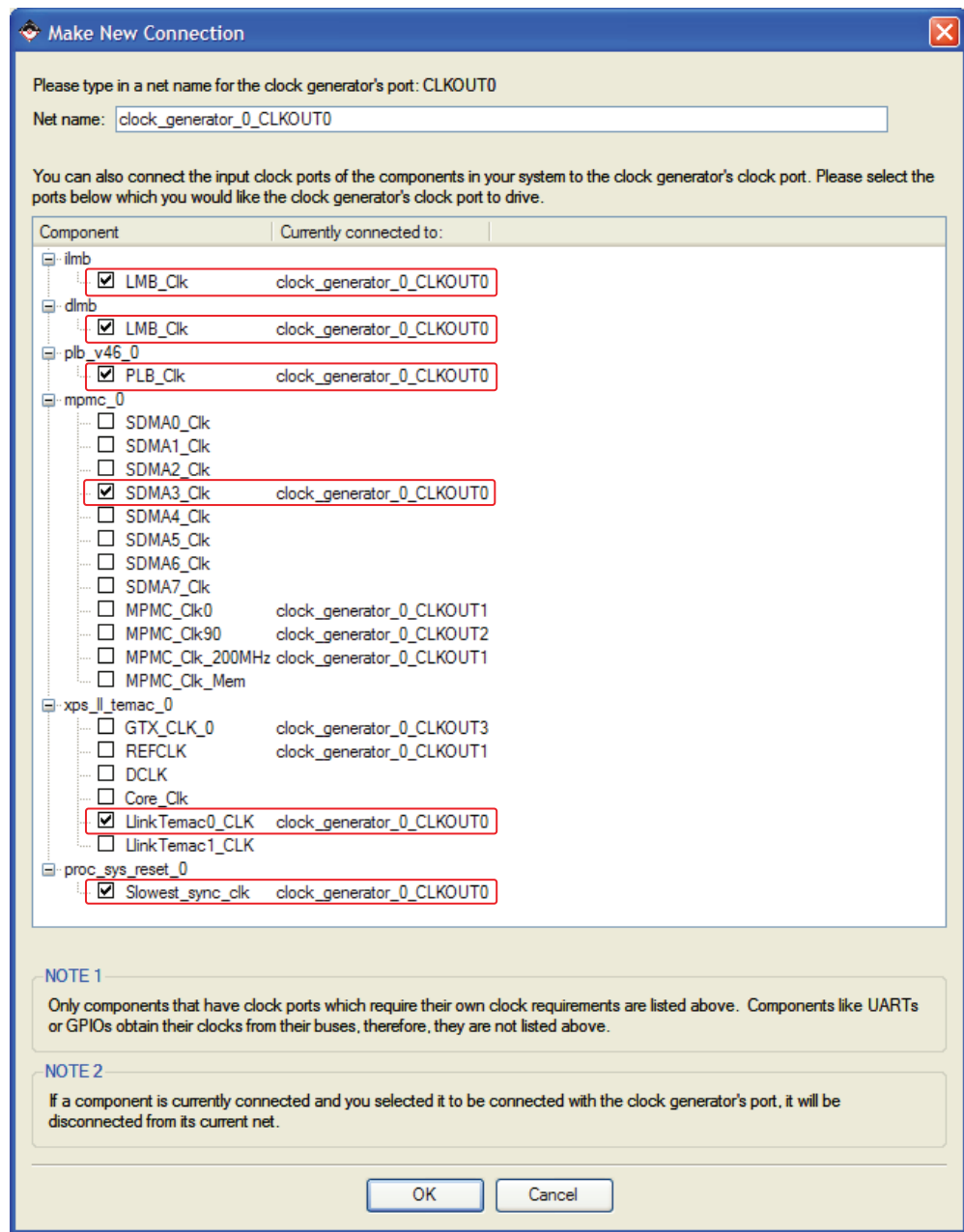
Select **LMB_Clk** for both **ilmb** and **dlmb**.

Select **SDMA3_Clk** under **mpmc_0**. This sets the SDMA clock.

Select **LlinkTemac0_CLK** under **xps_ll_temac_0** which selects the LocalLink clock.

Select **Slowest_sync_clk** under **proc_sys_reset_0**.

The above connections are shown in [Figure 4-21](#).



UG443_3_21_082007

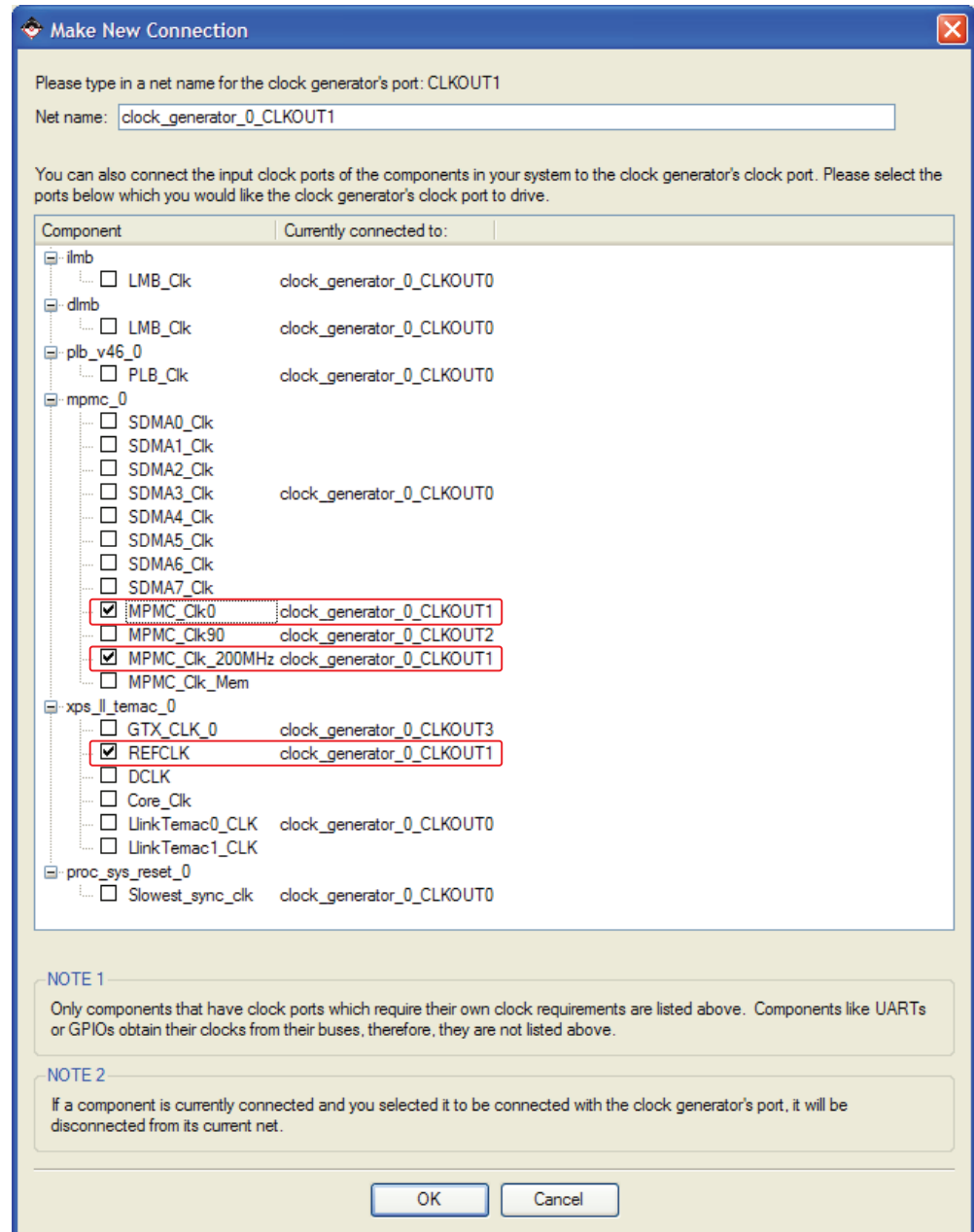
Figure 4-21: Clock Generator CLKOUT0 Connections

Under the **Ports** expand the **Outputs** tree node and click on **CLKOUT1**. On the right of the window, set the **Required frequency:** to **200000000 Hz**, **Required phase shift :** to **0**, and **Grouping Information:** to **Group0**. Then select **Connect to: New connection...**

Select **MPMC_Clk_200MHz** and **MPMC_Clk0** under **mpmc_0**.

Select **REF_CLK** under **xps_II_temac_0**.

The above connections are shown in [Figure 4-22](#).



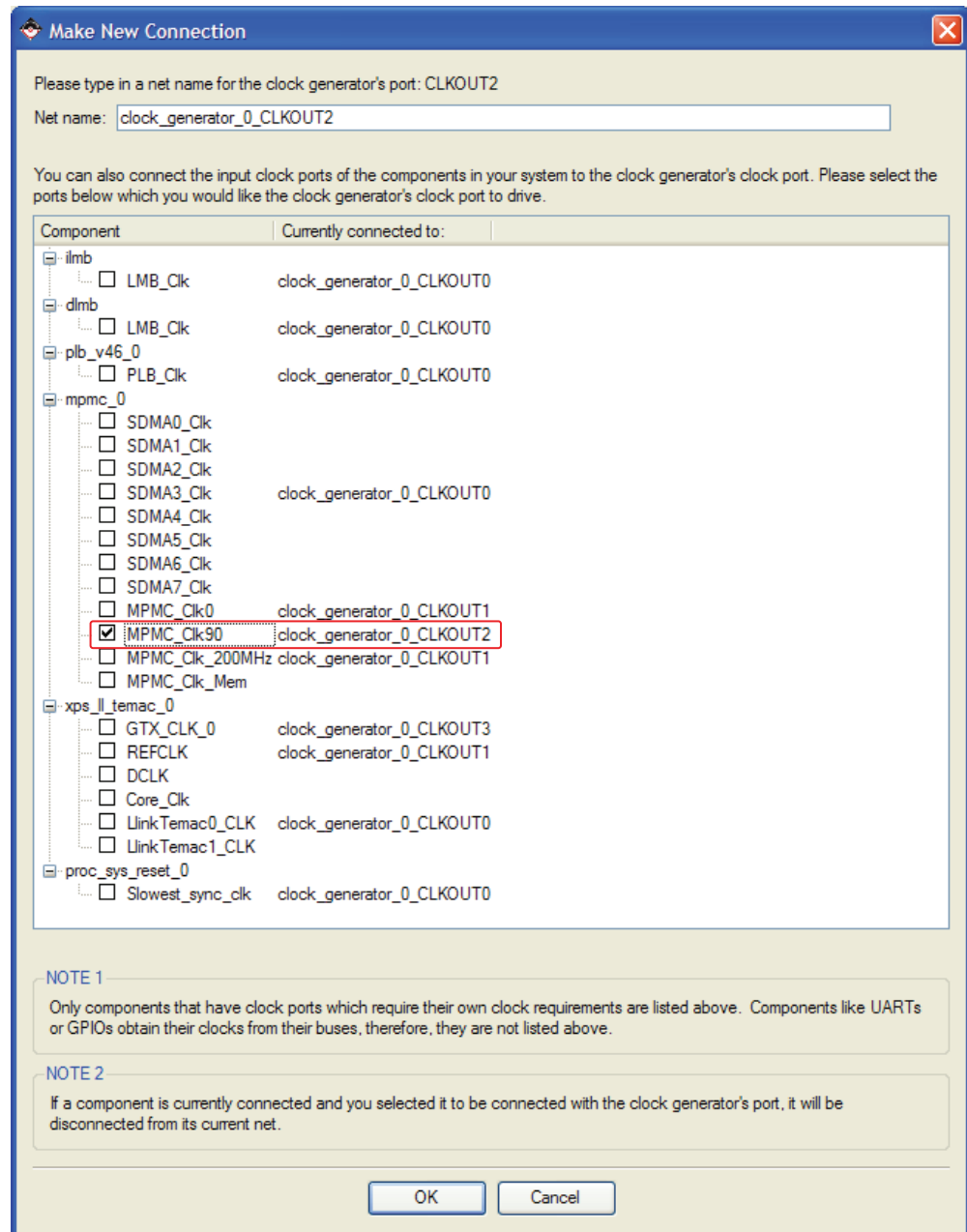
UG443_3_22_082007

Figure 4-22: Clock Generator CLKOUT1 Connections

Under the **Ports** expand the **Outputs** tree node and click on **CLKOUT2**. On the right of the window, set the **Required frequency**: to **200000000 Hz**, **Required phase shift** : to **90**, and **Grouping Information**: to **Group0**. Then select **Connect to: New connection...** .

Select **MPMC_Clk90** under **mpmc_0**.

The above connections are shown in [Figure 4-23](#).



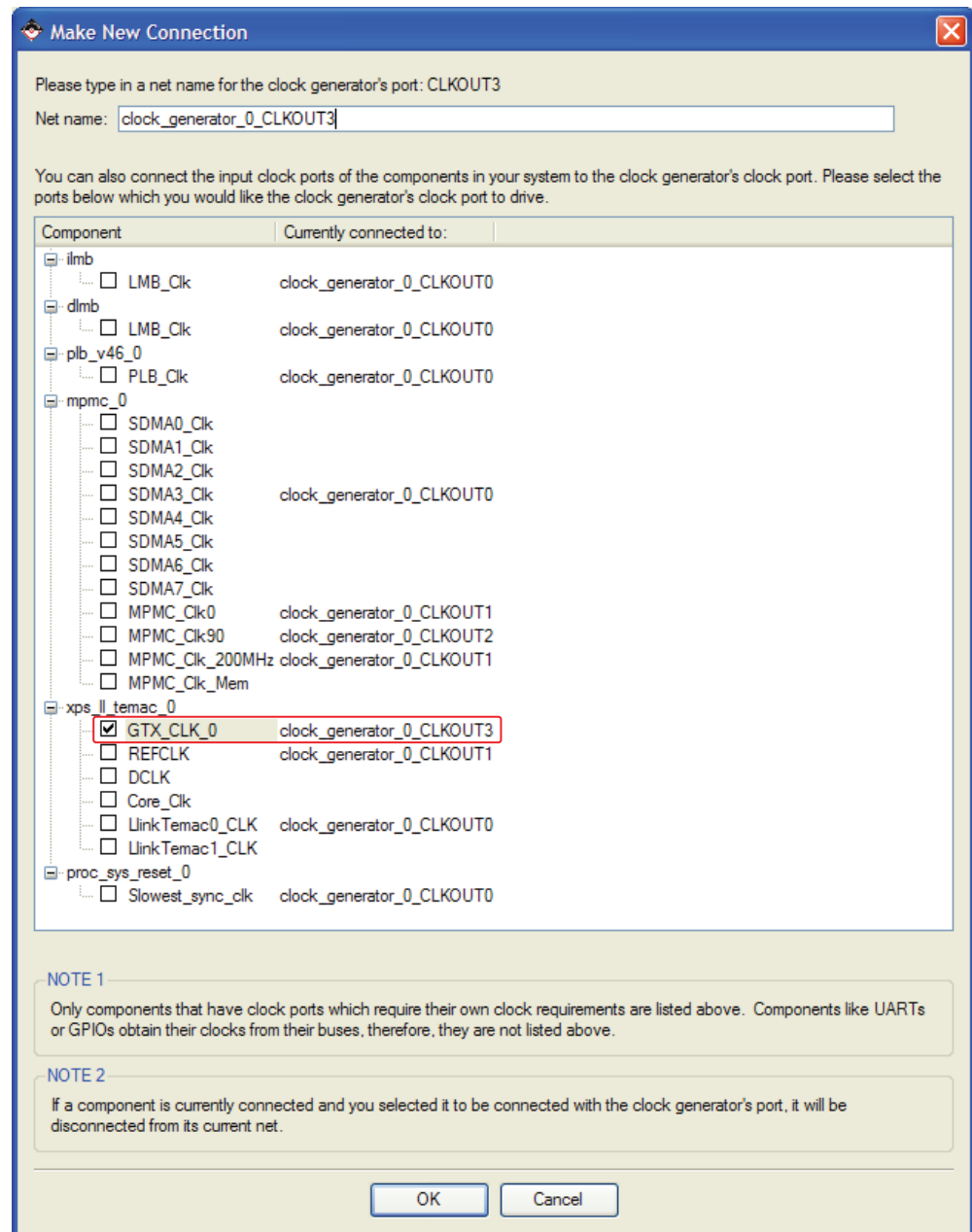
UG443_3_23_082007

Figure 4-23: Clock Generator CLKOUT2 Connections

Under the **Ports** expand the **Outputs** tree node and click on **CLKOUT3**. On the right of the window, set the **Required frequency**: to **125000000 Hz**, **Required phase shift**: to **0**, and **Grouping Information**: to **NONE**. Then select **Connect to: New connection...**

Select **GTX_CLK** under **xps_ll_temac_0**.

The above connections are shown in [Figure 4-24](#).



UG443_3_24_082007

Figure 4-24: Clock Generator CLKOUT3 Connections

Migration of User IP Slave Cores

Introduction

This chapter describes the process for migrating from an existing OPB or PLB v3.4 slave core to a PLB v4.6 slave core and retain the same services and functionality.

The following migration steps will be discussed:

1. Review the Create IP Wizard slave services available for the OPB/PLB v3.4 slave core
2. Review the Create IP Wizard slave services available for the PLB v4.6 slave core
3. Describe services and functionality of the original OPB/PLB v3.4 core created by Create IP Wizard
4. Use Create IP Wizard, within the EDK tools, to create an equivalent PLB v4.6 pcore for both PLB v3.4 and OPB slave cores
 - a. Set up services to match the functionality of the original OPB/PLB v3.4 slave core
 - b. Describe parameters dealing with PLB v4.6 slave pcores
5. Modify the existing user logic of the pcore to interface with the top level template created by Create IP Wizard

Overview of Create IP Wizard for OPB/PLBV34 Slaves

Create IP Wizard Slave Services

The slave services for OPB/PLBV34 are S/W reset and MIR, User Logic Interrupt Support, User Logic S/W register support, Burst Transaction Support, FIFO, and User Logic Address range support.

For OPB, the slave interface has a data width of 32 bits. For PLBV34, the slave interface has a data width of 64 bits.

OPB and PLBV34 IPIFs

The OPB and PLBV34 IPIFs includes slave services and master services which are enabled or disabled through parameters inside the IPIF instance. The Create IP Wizard enables or disables these parameter based upon user selections inside the Wizard.

Overview of Create IP Wizard using PLB v4.6

Create IP Wizard Slave Services

The slave services are similar to the OPB and PLBV34 slave services. The services are Software Reset, Read/Write FIFO, Interrupt Control, User logic Software Register, and User Logic Memory Space. The user has a option of adding burst and cache-line support inside the slave. Burst and native data width are discussed later in this chapter.

PLBV46 Slave IPIFs

Neither the PLBV46 Slave Single IPIF nor the PLBV46 Slave Burst IPIF support a Soft Reset service, Local IP Interrupt services, RdFIFO services, WrFIFO services or master services. These services are automatically added through the Create IP Wizard in the top level template file created by the Wizard.

PLBV46 Slave IPIFs do not support aborts or indeterminate length bursts. However, optimizations may be possible inside the user logic based upon the Bus2IP_BurstLength signal inside the IPIC.

In the PLBV46 Slave Burst IPIF, when the write buffer is included, the IP2Bus_Error signal has no affect on the bus. This is because the PLBV46 IPIF will acknowledge the data phase on the PLB immediately after the address phase, therefore not allowing the IP time to assert the IP2Bus_Error signal. To catch an IP error during a write transaction, the user must set an interrupt .

The selection of the PLBV46 Slave IPIF depends on the selection of Burst and cache-line support in the Wizard. If Burst and cache-line support is not enabled in the wizard, the PLBV46 Slave Single IPIF is used in the slave core. If Burst and cache-line support is enabled, the PLBV46 Slave Burst IPIF is used in the slave core.

Overview of OPB/PLBV34 Slave Cores Created by Create IP Wizard

The PLBV34 and OPB Slave Example pcores were created with Create IP Wizard, within the EDK tools. Both slave examples use the software reset and MIR support, user logic interrupt support with Device ISC (interrupt source controller) is enabled, user logic Software register support, RdFIFO and WrFIFO with Packet Mode and Vacancy Calculator, and burst transaction support. The setup of these services are the same for the OPB and PLBV34.

For OPB, the write mode to this core disables posted write behavior for normal acknowledge write behavior.

For OPB, four 32-bit registers are used in the user logic.

For PLBV34, four 64-bit registers are used in the user logic.

Creating PLBV46 Slave Cores with Create IP Wizard

Generating The PLBV46 Slave Cores Using Create IP Wizard

The steps in the following section are applied in generating both the migrated OPB and PLBV34 slave cores. Notice the migration considerations for both OPB/PLBV34 in the following steps.

1. Invoke the Create IP Wizard and click on **Next** as shown in [Figure 5-1](#).

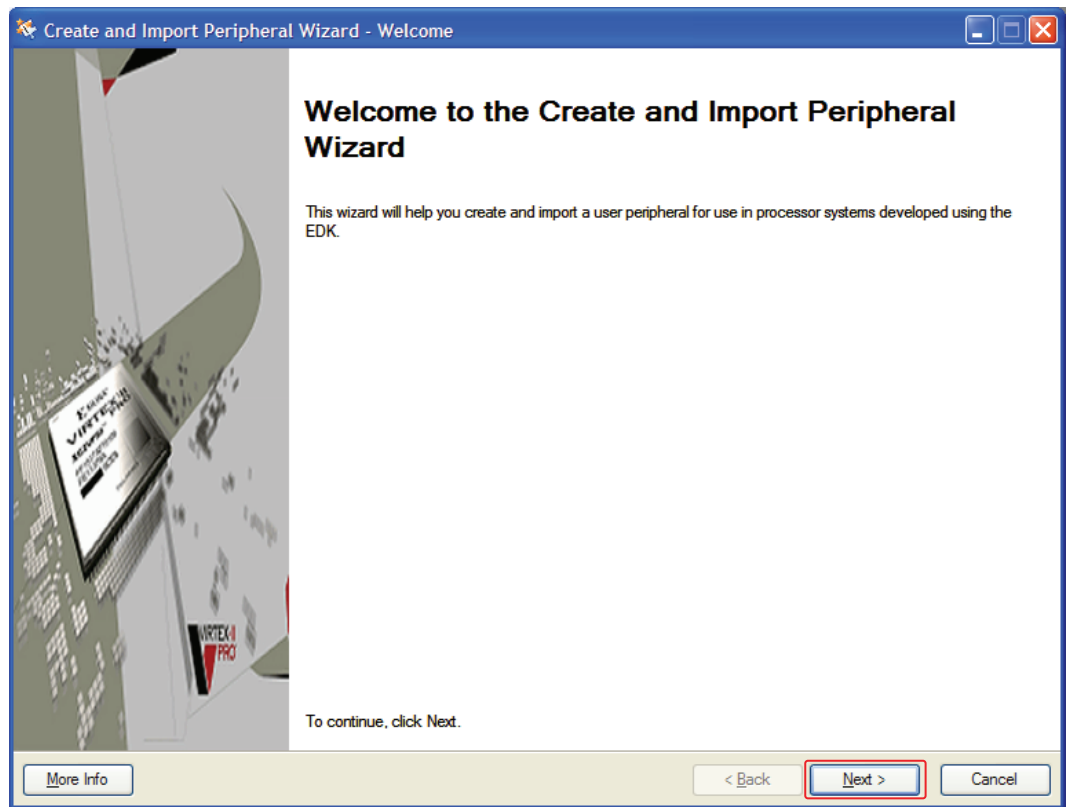


Figure 5-1: Create/IP Wizard Welcome

2. In the Create and Import Peripheral Wizard - Peripheral Flow window, under Select Flow, click on **Create templates for a new peripheral** as shown in Figure 5-2, then click on **Next**.

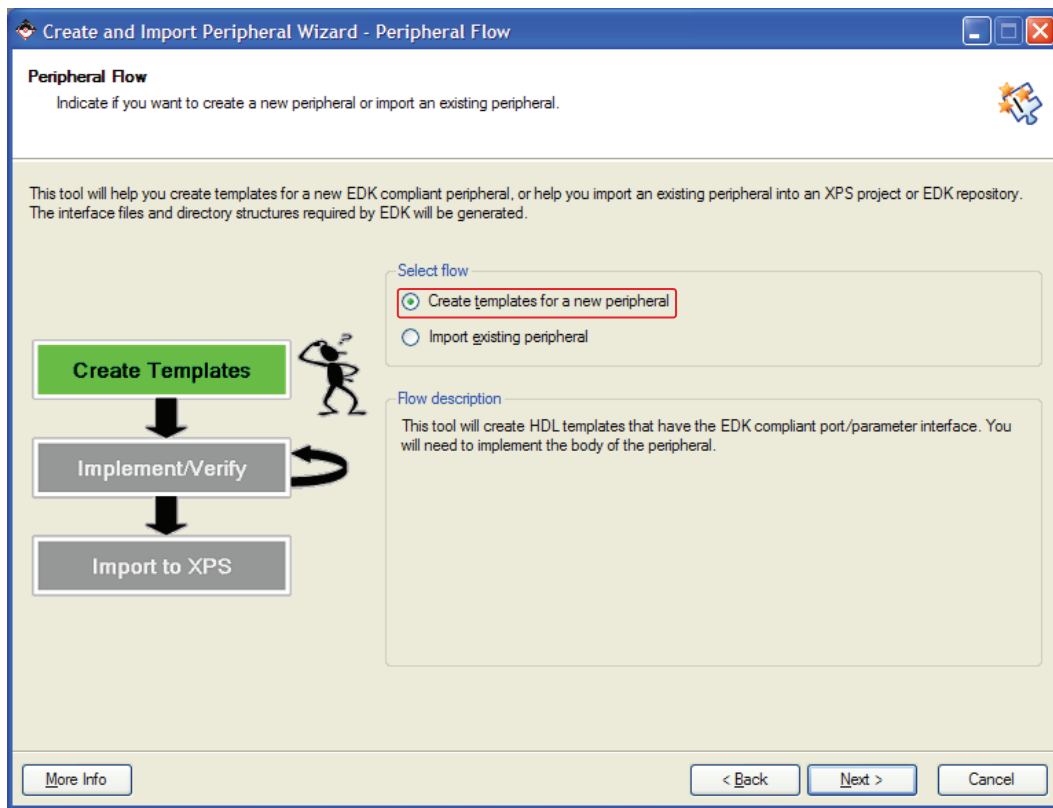
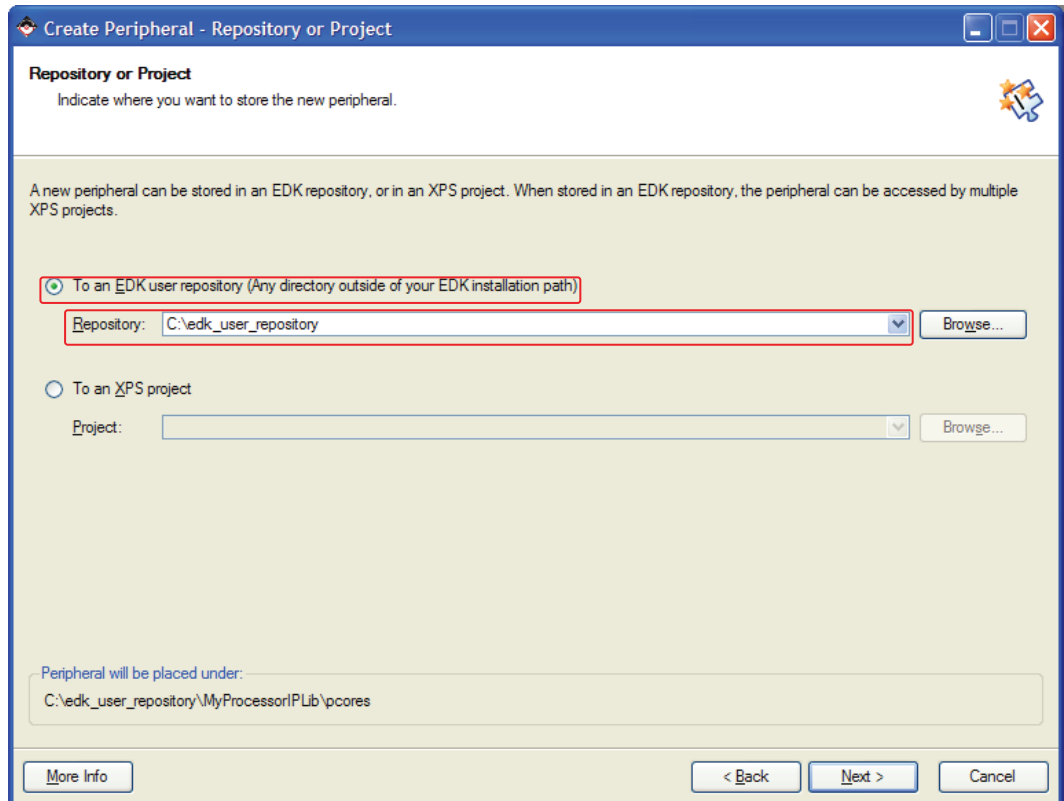


Figure 5-2: Create/IP Wizard Peripheral Flow

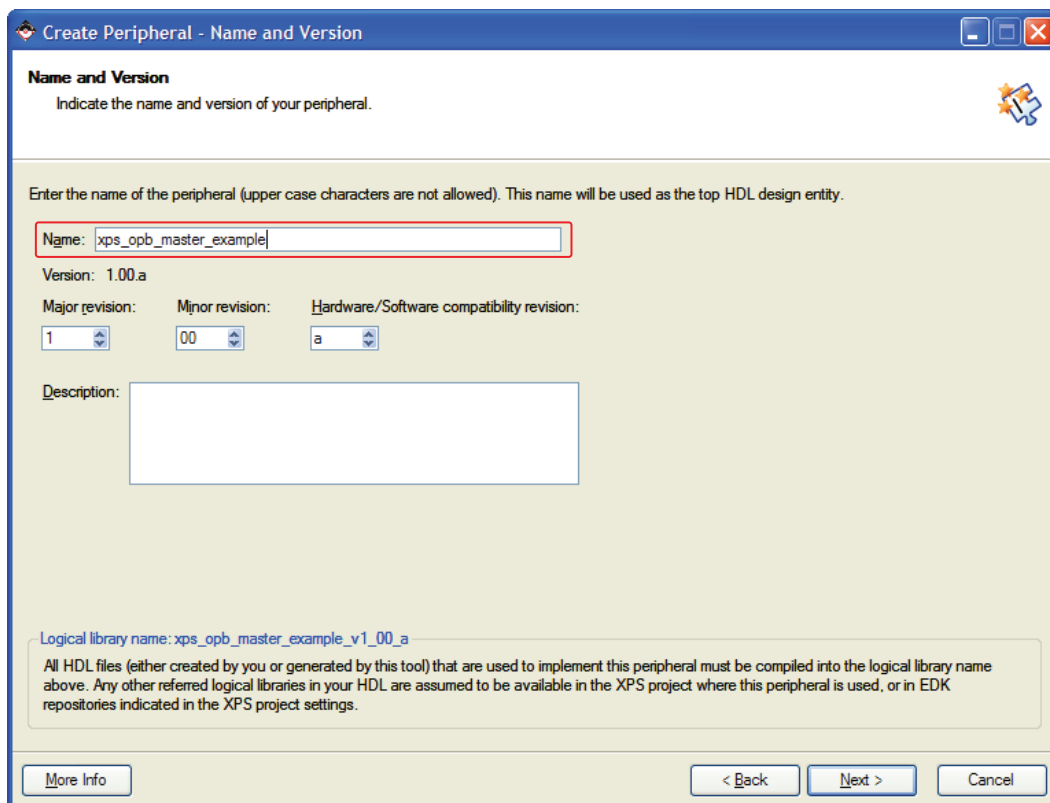
3. In the Create Peripheral - Repository or Project window, select a location for the repository where the cores will be generated as shown in [Figure 5-3](#), then click on **Next**.



UG443_4_3_082007

Figure 5-3: Create/IP Wizard Repository

4. In the Create Peripheral - Name and Version window, enter the name of the slave migrated core that will be created as shown in [Figure 5-4](#) for the OPB migration, then click on **Next**.



Create Peripheral - Name and Version

Name and Version
Indicate the name and version of your peripheral.

Enter the name of the peripheral (upper case characters are not allowed). This name will be used as the top HDL design entity.

Name:

Version: 1.00.a

Major revision: Minor revision: Hardware/Software compatibility revision:

Description:

Logical library name: xps_opb_master_example_v1_00_a

All HDL files (either created by you or generated by this tool) that are used to implement this peripheral must be compiled into the logical library name above. Any other referred logical libraries in your HDL are assumed to be available in the XPS project where this peripheral is used, or in EDK repositories indicated in the XPS project settings.

UG443_5_4_082007

Figure 5-4: Create/IP Wizard Name and Version for OPB Migration

5. In the Create Peripheral - Name and Version window, enter the name of the slave migrated core that will be created for the PLB migration example as shown in [Figure 5-5](#), then click on **Next**.

Create Peripheral - Name and Version

Name and Version
Indicate the name and version of your peripheral.

Enter the name of the peripheral (upper case characters are not allowed). This name will be used as the top HDL design entity.

Name:

Version: 1.00.a

Major revision: Minor revision: Hardware/Software compatibility revision:

Description:

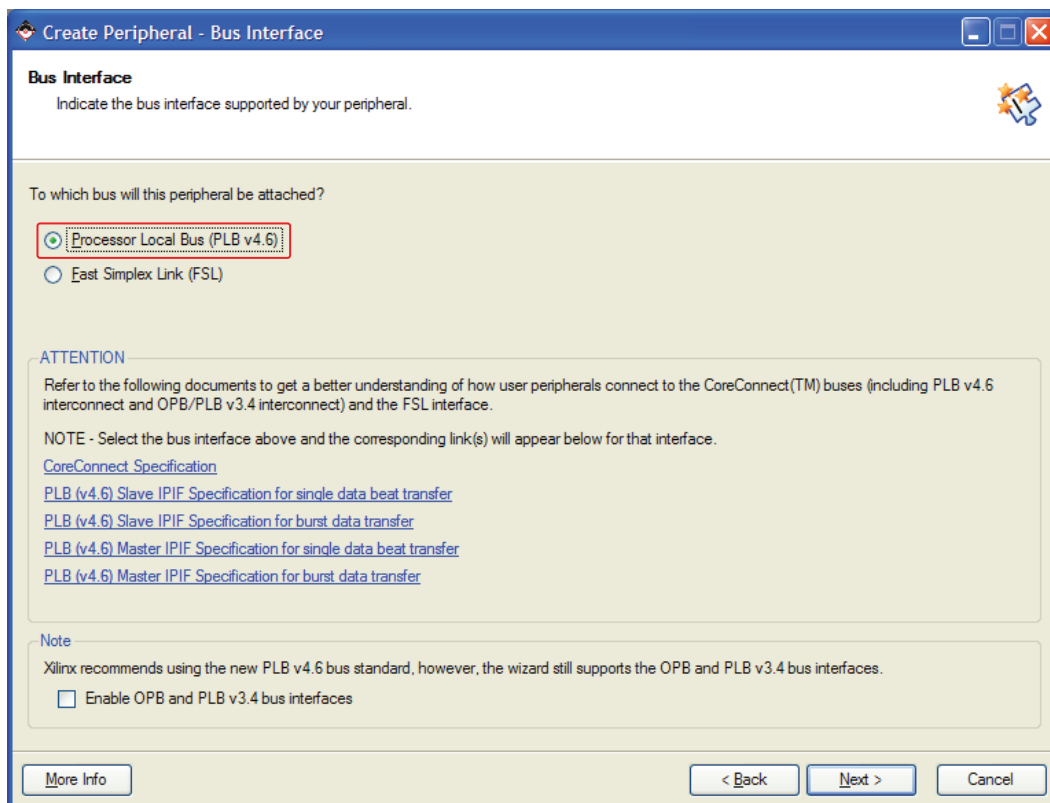
Logical library name: xps_plb_master_example_v1_00_a

All HDL files (either created by you or generated by this tool) that are used to implement this peripheral must be compiled into the logical library name above. Any other referred logical libraries in your HDL are assumed to be available in the XPS project where this peripheral is used, or in EDK repositories indicated in the XPS project settings.

UG443_4_5_082007

Figure 5-5: Create/IP Wizard Name and Version for PLB Migration

6. In the Create Peripheral - Bus Interface window, under **To which bus will this peripheral be attached?**, click on **Processor Local Bus (PLB v4.6)** as shown in Figure 5-6, then click on **Next**.

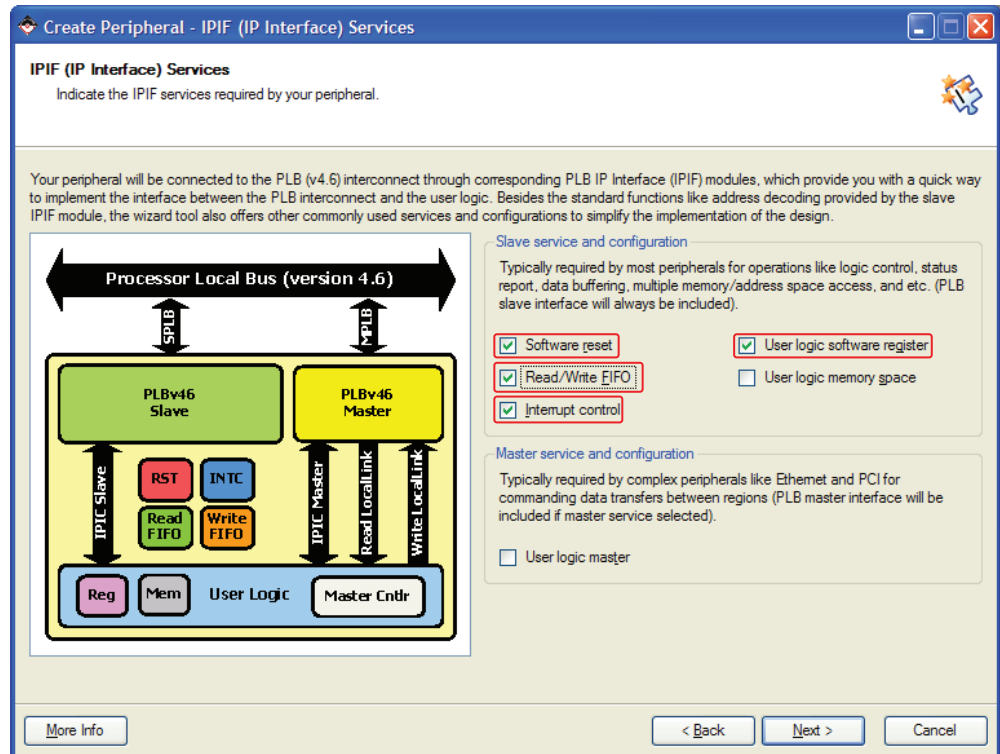


UG443_4_6_082007

Figure 5-6: Create/IP Wizard Bus Interface

- In the Create Peripheral - IPIF (IP Interface) Services window, select the services, **Software reset**, **Read/Write FIFO**, **Interrupt control**, and **User logic software register** as shown in Figure 5-7, then click on **Next**.

Note that no master services are selected.



UG443_4_7_082007

Figure 5-7: Create/IP Wizard IPIF Services

8. In the Create Peripheral - Slave Interface window, several design considerations are needed for OPB and PLBV34.

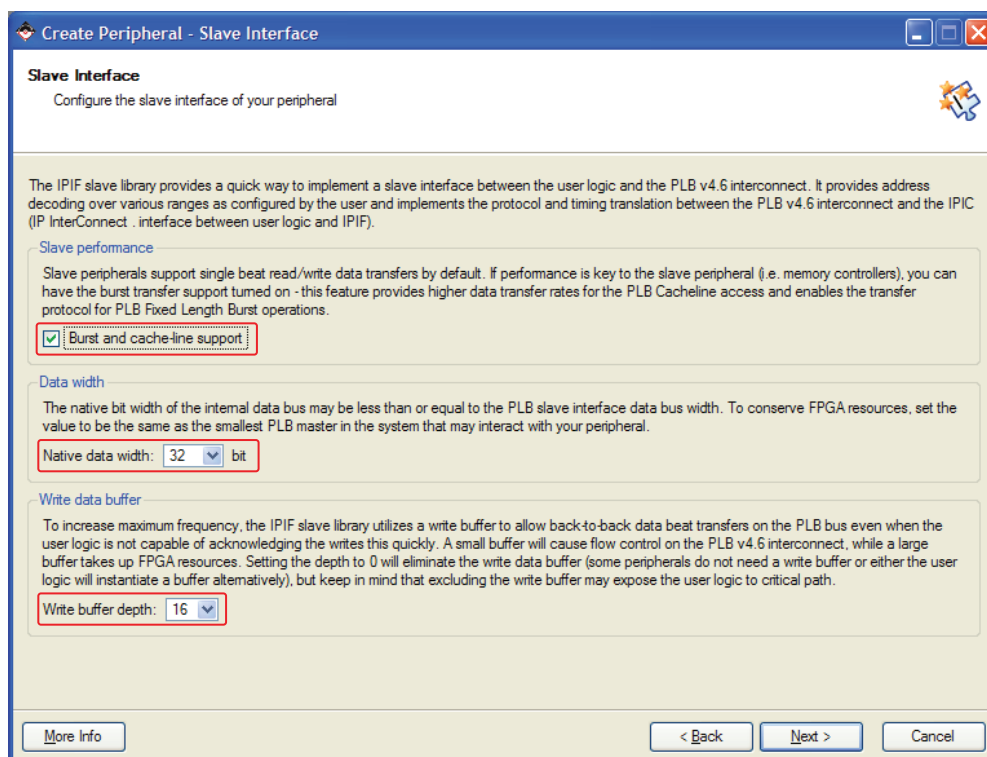
If Burst and cache-line support is not enabled, the native data width is fixed at 32 bits, and bursting is not available. If Burst and cache-line support is enabled, the native data width is variable at 32, 64, and 128 bits, and bursting is available.

If the Smallest Master Native Dwidth < Slave Native Dwidth, byte steering logic is included which takes additional resources.

- ◆ OPB

In a typical OPB system the largest master has a data width of 32 bits. In this case, the user has an option of enabling or disabling Burst and cache-line support because the data width of the bus is 32 bits. The user should base this decision on the functionality of the core. If Burst and cache-line support is enabled, set the inclusion of the Write data buffer.

For this example, check **Burst and cache-line support**, set the **Native data width:** to 32 bit, and set the **Write buffer depth:** to 16 as shown in Figure 5-8, then click on **Next**.



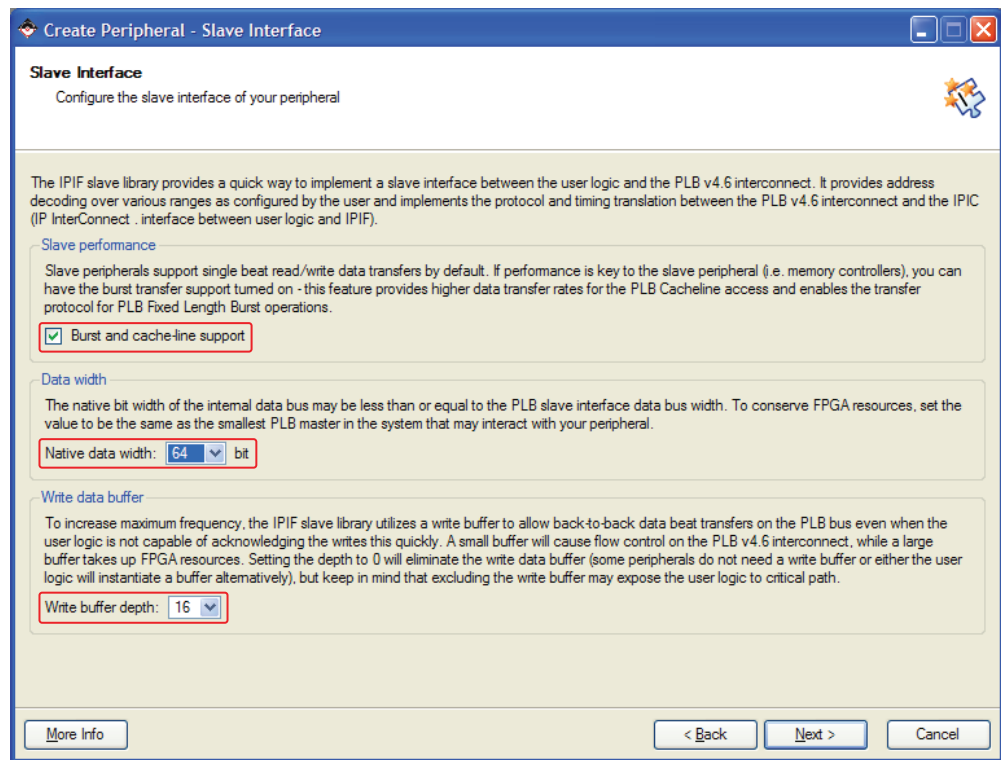
UG443_4_8_082007

Figure 5-8: Create/IP Wizard Slave Interface for OPB Migration

◆ PLBV34

In a typical PLBV34 system the largest master has a data width of 64 bits. In this case, check **Burst and cache-line support**. The user has the option of setting the Native data width based upon needs of the system. In addition, the user can set the Write data buffer.

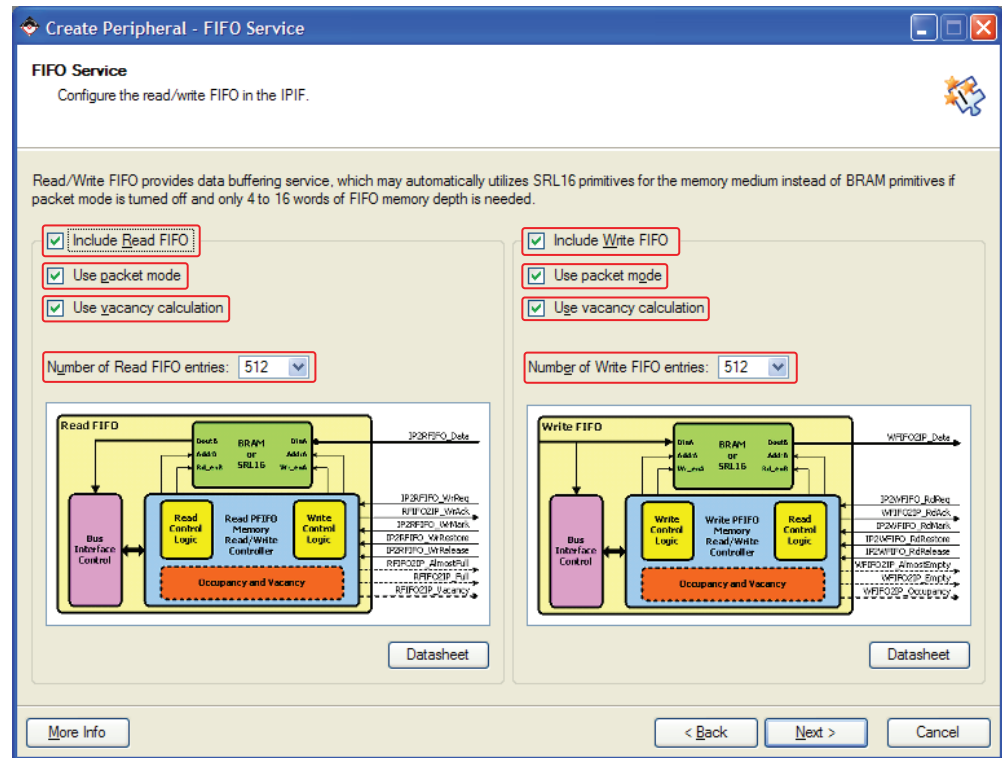
In this example, set the **Native data width:** to **64** bits and the **Write buffer depth:** to **16** as shown in [Figure 5-9](#), then click on **Next**.



UG443_4_9_082007

Figure 5-9: Create/IP Wizard Slave Interface for PLB Migration

9. In the Create Peripheral - FIFO Service window, make the selections shown in Figure 5-10 to configure the read/write FIFO, then click on **Next**.

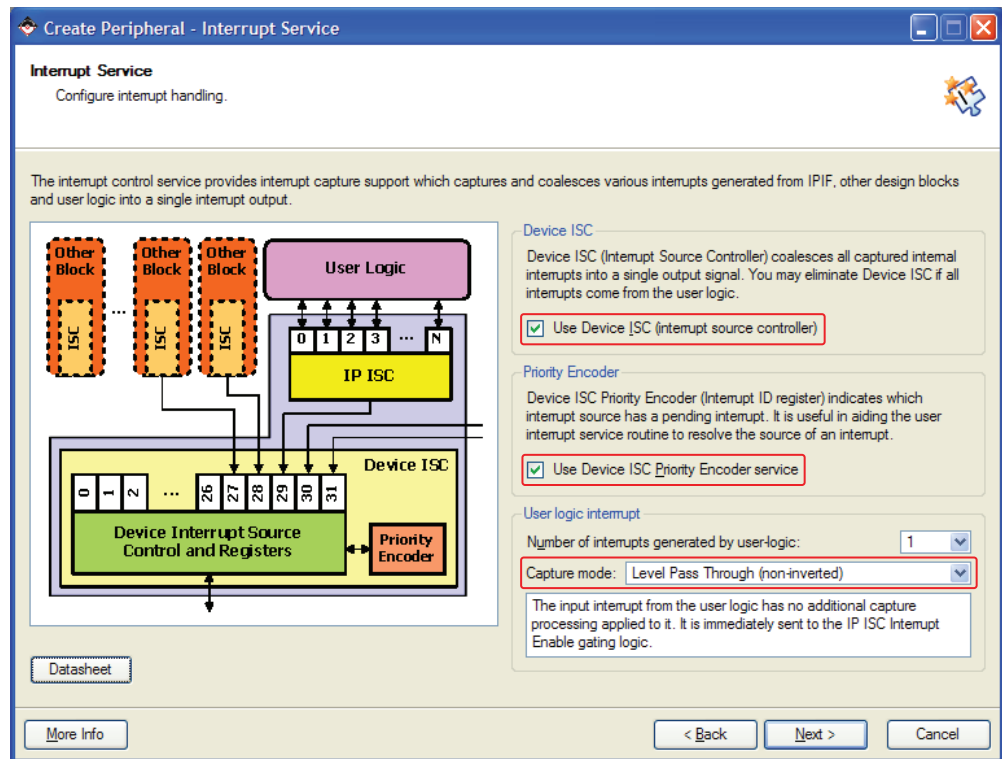


UG443_4_10_082007

Figure 5-10: Create/IP Wizard FIFO Service

- In the Create Peripheral - Interrupt Service window, check **Use Device ISC (Interrupt source controller)** and **Use Device ISC Priority Encoder service**. Select **Level Pass Through (non-inverted)** as the **Capture Mode**, and **1** as the **Number of interrupts generated by user logic**: as shown in Figure 5-11:

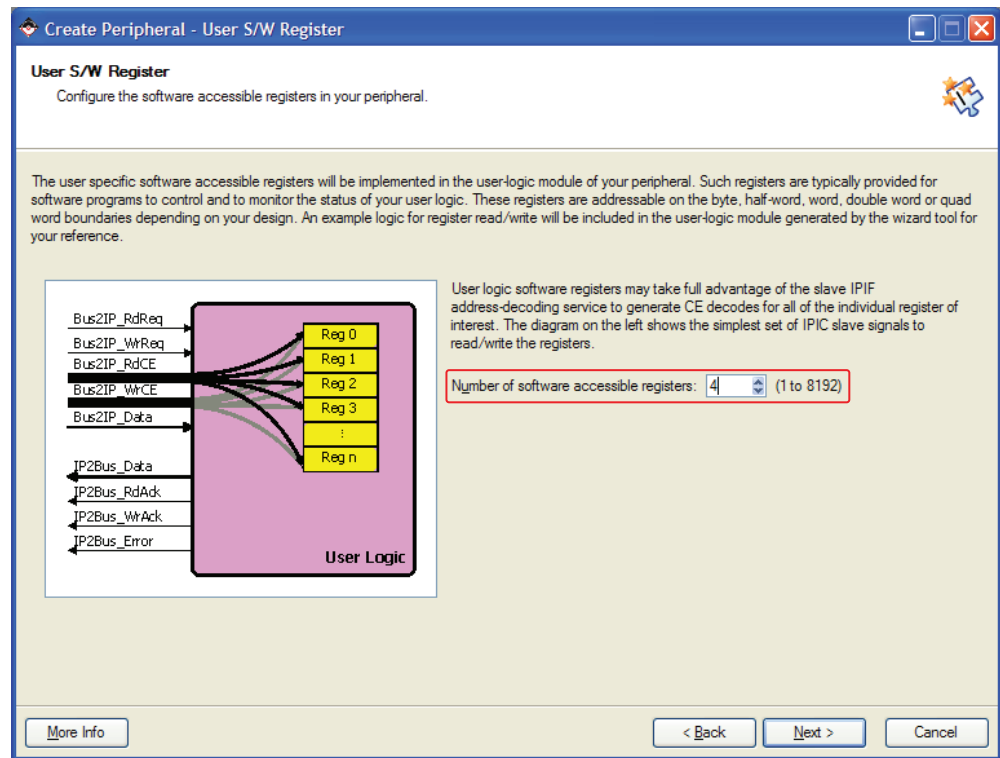
Click on **Next**.



UG443_4_11_082007

Figure 5-11: Create/IP Wizard Interrupt Service

- In the Create Peripheral - User S/W Register window, set the **Number of software accessible registers:** to 4 as shown in Figure 5-12, then click on **Next**.



UG443_4_12_082007

Figure 5-12: Create/IP Wizard User S/W Register

- In the Create Peripheral - IP Interconnect (IPIC) window shown in Figure 5-13, note the standard IPIC signals connected to the user logic that have been automatically selected by the Wizard, then click on **Next**.

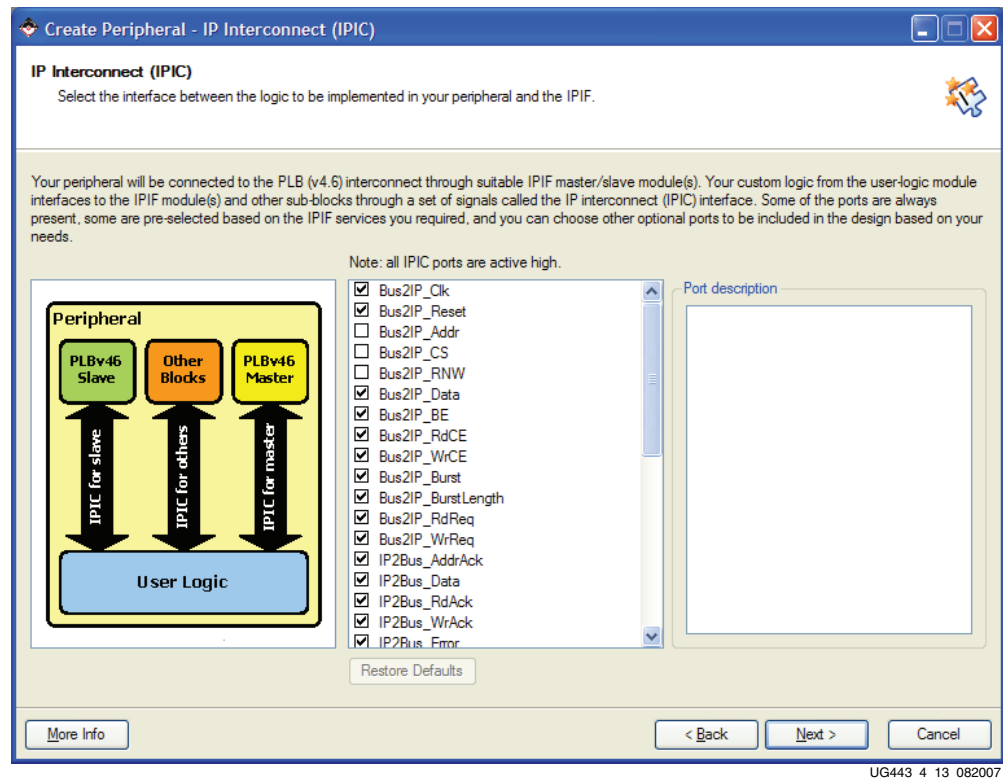
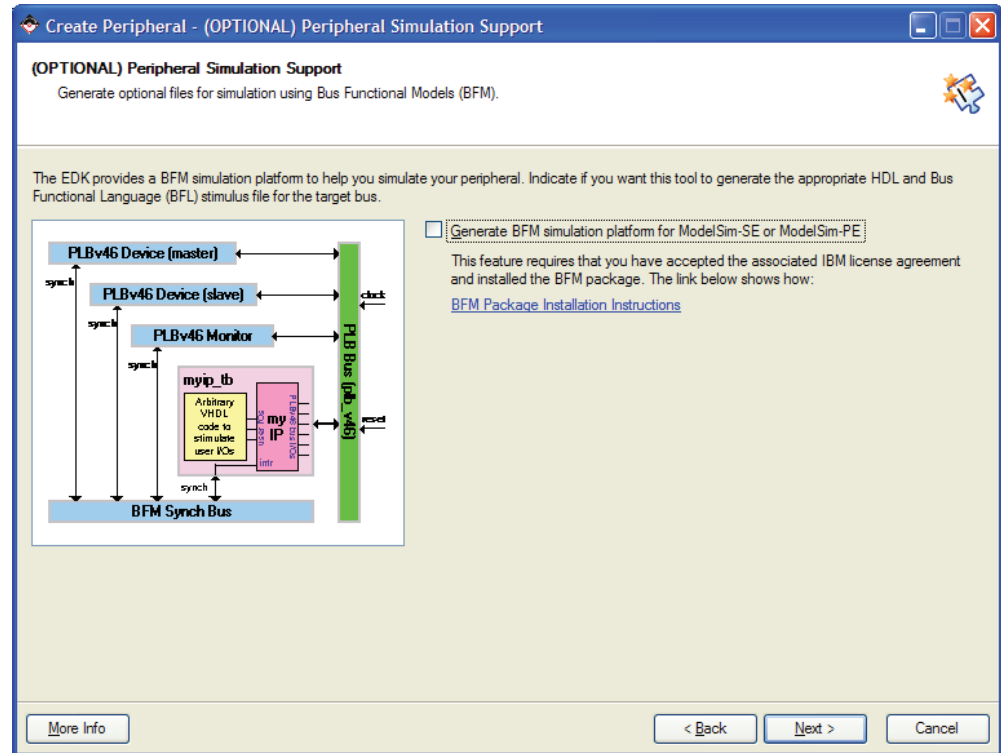


Figure 5-13: Create/IP Wizard IP Interconnect (IPIC)

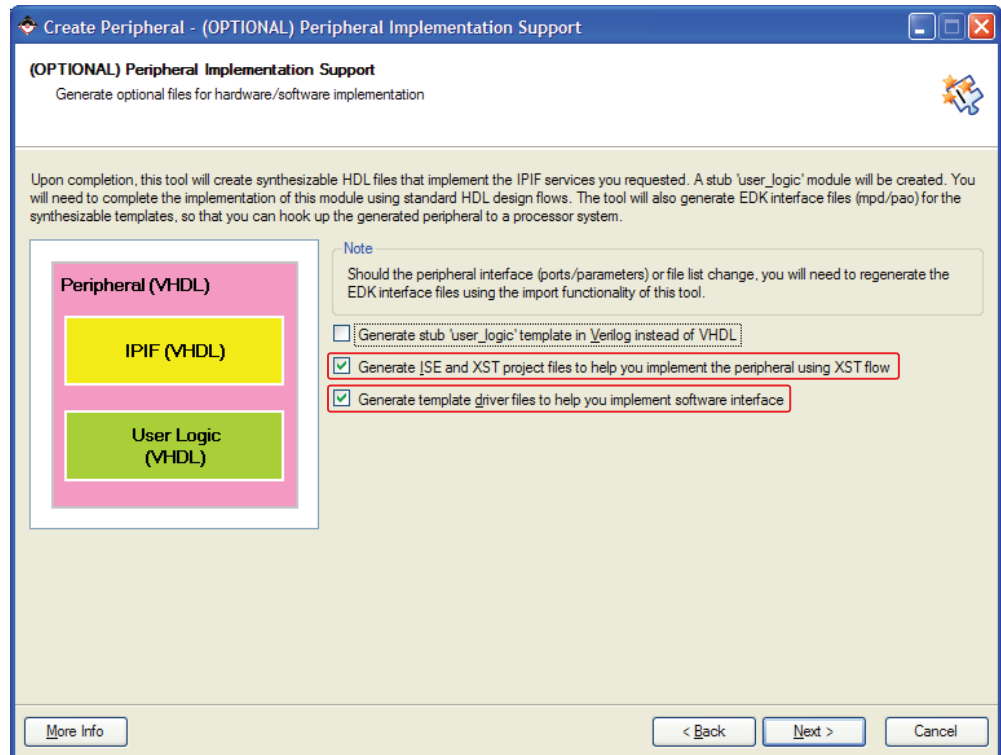
13. In the Create Peripheral - (OPTIONAL) Peripheral Simulation Support window, the user has the the option of clicking on **Next** to continue or selecting **Generate BFM simulation platform for ModelSim-SE or ModelSim-PE** to create a BFM environment to test the core.



UG443_4_14_082007

Figure 5-14: Create/IP Wizard Peripheral Simulation Support

14. In the Create Peripheral - (OPTIONAL) Peripheral Implementation Support shown in [Figure 5-15](#), check **Generate ISE and XST project files to help you implement the peripheral using XST flow** and **Generate template driver files to help you implement software interface**, then click on **Next**.



UG443_4_15_082007

Figure 5-15: Create/IP Wizard (OPTIONAL) Peripheral Implementation Support

15. In the Create Peripheral - Finish window shown in [Figure 5-16](#), click on **Finish** to generate the core.

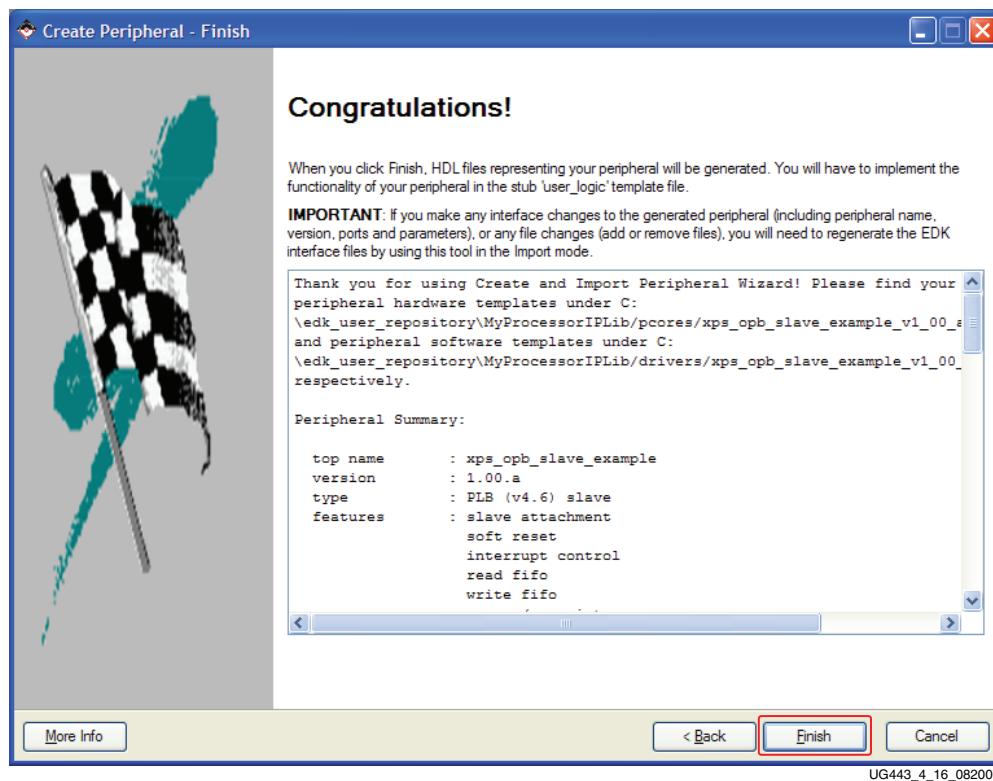


Figure 5-16: Create/IP Wizard Finish

Parameters for PLBV46 Slave Cores

Certain parameters for the PLBV46 Slave interface are necessary for the tools as shown in [Table 5-1](#).

Note: Some parameters in [Table 5-1](#) are overwritten inside EDK to the system values depending on the system.

Table 5-1: PLBV46 Slave Burst and PLBV46 Slave Single Required Parameters

Parameter	Description	Allowable Values
C_SPLB_AWIDTH	Width of the PLB Address Bus.	32
C_SPLB_DWIDTH	Width of the PLB Data Bus.	32,64,128
C_SPLB_NATIVE_DWIDTH	Width of the slave IPIF data bus.	32,64,128
C_SPLB_NUM_MASTERS	Number of PLB Masters.	1 to 16
C_SPLB_MID_WIDTH	PLB Master ID Bus Width.	1 to 4
C_SPLB_SUPPORTS_BURST	Slave supports burst.	0,1
C_SPLB_P2P	Selects point-to-point or shared PLB topology.	0,1
C_SPLB_SMALLEST_MASTER	Data width of smallest master that will access the IPIF.	32,64,128

Note: Usually the PLBV34 bus has a DWIDTH of 64 bits, while the OPB has a DWIDTH of 32 bits.

[Table 5-2](#) shows the parameters specific to the PLBV46 Slave Single IPIF entity in top level template.

Table 5-2: PLBV46 Slave Single Parameters

Parameter	Description	Allowable Values
C_SIPIF_DWIDTH	Same as C_SPLB_NATIVE_DWIDTH.	32
C_BUS2CORE_CLK_RATIO	Selects the ratio of bus clock to core clock for use in dual clock systems.	1,2

The user has an option of running the core at one half of the bus speed by setting the C_BUS2CORE_CLK_RATIO parameter to 2 (2:1 ratio). The user must provide a slower clock that is edge synchronized with the bus clock and not use the Bus2IP_Clk from the IPIF. The Create IP Wizard does not have this option when generating a slave core.

[Table 5-3](#) shows the parameters specific to the PLBV46 Slave Burst IPIF entity in the top level template.

Table 5-3: PLBV46 Slave Burst Parameters

Parameter	Description	Allowable Values
C_SIPIF_DWIDTH	Same as C_SPLB_NATIVE_DWIDTH.	32,64,128
C_WR_BUFFER_DEPTH	Size of the write buffer.	0,16,32,64
C_CACHELINE_ADDR_MODE	Cache Link addressing mode.	0,1

The `C_WR_BUFFER_DEPTH` parameter determines the size of the write buffer. If the parameter is set to `0`, the buffer is removed. When a buffer exists, (only with the PLBV46 Slave Burst), special consideration is needed when dealing with IPIC errors.

Modifying the Existing User Logic

Modifications are needed to the existing OPB and PLBV34 User Logic source VHDL files to match the top template generated by the Create IP Wizard. A block diagram of this process is shown in [Figure 5-17](#).

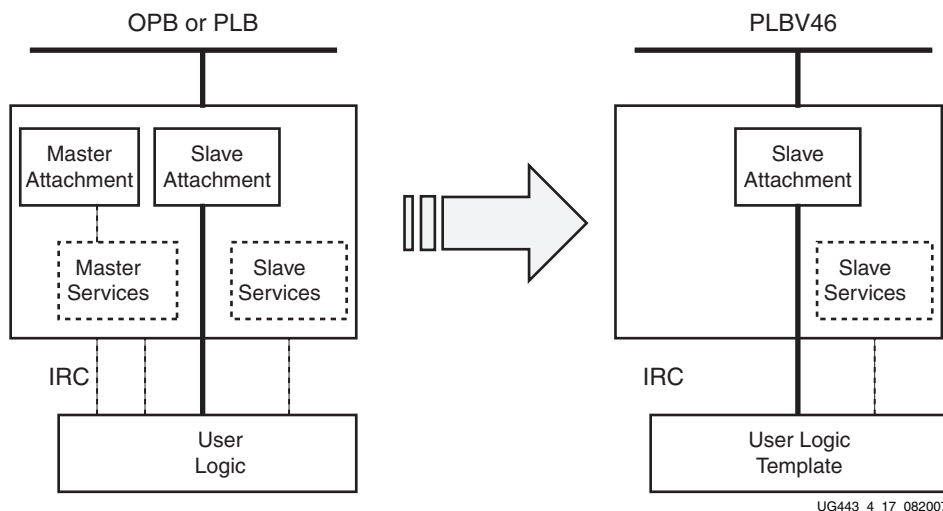


Figure 5-17: Modifying User Logic

Modifying Existing OPB User Logic

The entity of the existing user logic is modified to match the top level template declaration of User Logic.

Modifying User Logic Entity Generics

The following parameters are globally replaced in the existing User Logic.

`C_DWIDTH` -> `C_SLV_DWIDTH`

`C_NUM_CE` -> `C_NUM_REG`

`C_IP_INTR_NUM` -> `C_NUM_INTR`

Remove the parameters, `C_RDFIFO_DWIDTH` and `C_WRFIFO_DWIDTH`, from the entity. The other instances in the User Logic are replaced with `C_SLV_DWIDTH` in both cases.

Modifying User Logic Entity Ports and Connecting Logic

The following ports are added to the entity:

```
Bus2IP_BurstLength      : in  std_logic_vector(0 to 8);
Bus2IP_RdReq            : in  std_logic;
Bus2IP_WrReq            : in  std_logic;
IP2Bus_RdAck            : out std_logic;
IP2Bus_WrAck            : out std_logic;
```

The following port is removed because there are separate acknowledgement signals for read and write.

```
IP2Bus_Ack              : out std_logic;
```

The following connecting logic is removed:

```
IP2Bus_AddrAck <= slv_write_ack or slv_read_ack;
```

The removed line is replaced with:

```
IP2Bus_WrAck <= slv_write_ack;
IP2Bus_RdAck <= slv_read_ack;
```

Modifying Existing PLBV34 User Logic

The entirety of the existing user logic is modified to match the top level template declaration of User Logic.

Modifying User Logic Entity Generics

The following parameters are globally replaced inside the existing User Logic

C_DWIDTH -> C_SLV_DWIDTH

C_NUM_CE->C_NUM_REG

C_IP_INTR_NUM->C_NUM_INTR

Remove the following parameters from the entity. C_RDFIFO_DWIDTH and C_WRFIFO_DWIDTH. The other instances inside the User Logic are replaced with C_SLV_DWIDTH in both cases.

Modifying User Logic Entity Ports and Connecting Logic

The following ports are added to the entity:

```
Bus2IP_BurstLength      : in  std_logic_vector(0 to 8);
```

The follow ports are removed from the entity:

```
IP2Bus_ToutSup          : out std_logic;
IP2Bus_Busy              : out std_logic;
IP2Bus_Retry             : out std_logic;
```

The following connecting logic is removed:

```
IP2Bus_Busy             <= '0';
IP2Bus_Retry            <= '0';
IP2Bus_ToutSup          <= '0';
```


Migration of User IP Master/Slave Cores

Introduction

This chapter describes the migration process to go from an existing OPB or PLBv 3.4 master/slave core to a PLB v4.6 master/slave core.

The following migration steps will be discussed:

1. Review the Create IP Wizard master/slave services available for OPB/PLB v3.4 cores
2. Review the Create IP Wizard master/slave services available for PLB v4.6 cores
3. Describe services and functionality of the original OPB/PLB v3.4 master/slave cores created by Create IP Wizard
4. Use Create IP Wizard, within the EDK tools, to create an equivalent PLB v4.6 core for both PLB v3.4 or OPB master/slave cores
 - a. Set up services to match the functionality of the original OPB/PLB v3.4 master/slave core
 - b. Describe parameters dealing with the PLB v4.6 master/slave pcores

Overview of Create IP Wizard for OPB/PLBV34 Master/Slaves

Create IP Wizard Master Services

Create IP Wizard allows for the user to select DMA and User logic master support for both OPB and PLB v3.4. The wizard then creates an example logic inside the User Logic to demonstrate Master transactions.

For OPB, the master and slave interfaces has a data width of 32-bits. For PLB v3.4, the master and slave interfaces has a data width of 64-bits.

OPB/PLBV34 IPIFs

The OPB/PLBV34 IPIFs includes slave services and master services which are enabled/disabled through parameters inside the IPIF instance. Create IP Wizard enabled/disabled these parameter based upon user sections inside the Wizard.

Overview of Create IP Wizard for PLB v4.6 Master/Slaves

Create IP Wizard Master Services

Create IP Wizard offers the User logic master support service. The user has a option of adding Burst and cache-line support for the master interface. Burst and native data width of the Master are discussed later in this chapter.

The wizard then creates an example inside the User Logic logic showing the IPIC master interface receiving and transmitting data via the LocalLink interface.

Note: No DMA services are available through Create IP Wizard.

PLBV46 Master/Slave IPIFs

With PLBV46 IPIFs, master/slave cores require 2 separate PLBV46 Slave and PLBV46 Master IPIFs, for the master and slave interfaces for the core. For example, PLBV46 Slave Single can be used for slave registers to setup master transactions and the PLBV46 Master Burst can be used for the Master attachment.

Unlike the OPB/PLBV34 IPIFs, the master read/write transactions no longer involve the slave attachment. In turn, the master no longer supplies an IP2IP address.

With PLBV46 Master Burst, the User IP reads and writes from the PLB Master via the Xilinx LocalLink Interface protocol which is a different protocol than PLBV34/OPB IPIFs. The PLBV46 Master Single uses a simplified LocalLink Interface protocol for single reads and writes.

Indeterminate length bursts are not supported with PLBV46 Master Burst. The User Logic must only generate fixed length bursts of 2 to 16 data beats. The length is specified in bytes and a multiple of the Native Dwidth/8. In addition, transactions cannot be aborted. With unaligned addresses, the User IP has to issue single data beat requests until address alignment is established. Access to critical slave registers from the processor or another master that control Master operations must be considered since there is no IP2Bus_Busy signal from the slave.

The selection of the PLBV46 Master IPIF depends on the selection of Burst and cache-line support inside the Wizard. If Burst and cache-line support is not enabled inside the wizard, the PLBV46 Master Single IPIF is used for the master interface. If Burst and cache-line support is enabled, the PLBV46 Master Burst IPIF is used for the master interface.

No master services, like DMA, are included with the PLBV46 Master IPIFs. The user has the option of adding LocalLink interface(s) to the User Logic which can be connected to the Soft DMA(SDMA). DMA migration is described later inside the *Migration of DMA Solutions* chapter.

Overview of OPB/PLBV34 Master/Slave Cores Created By Create IP Wizard

The OPB and PLBV34 Master/Slave Example pcores were created with Create IP Wizard, within the EDK tools. Both example cores use DMA in Simple Mode and User logic master support while no slave services are selected. Even though the master support is selected, the OPB IPIF or the PLBV34 IPIF is set to include both a master and slave. The slave is used to access registers that control master transactions. In this configuration, the master will contain sixteen 8-bit registers for control and status.

Creating the PLBV46 Master/Slave Cores Using Create IP Wizard

Generating the PLBV46 Master/Slave Cores

Follow the steps in the following section to generate both the migrated OPB and PLBV34 master/slave cores. Notice the migration considerations for both OPB/PLBV34 in the following steps.

1. Invoke the Create IP Wizard and click on **Next** as shown in [Figure 6-1](#).

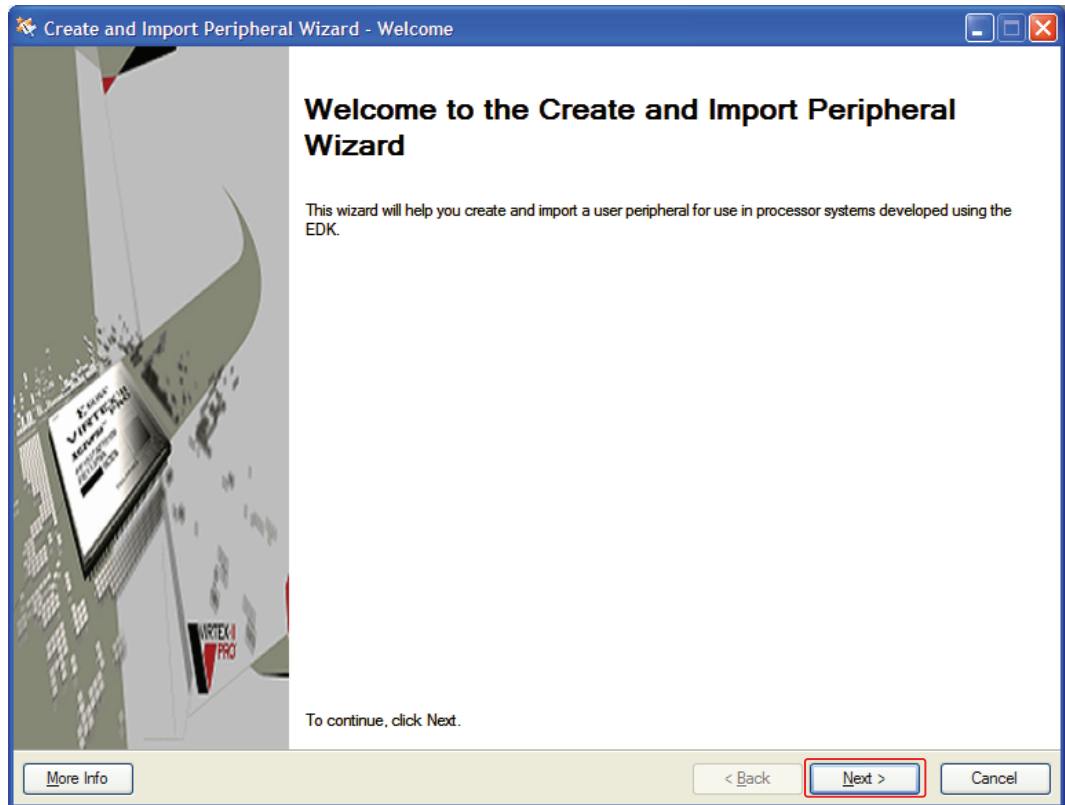


Figure 6-1: Create/IP Wizard Welcome

2. In the Create and Import Peripheral Wizard - Peripheral Flow window, under **Select flow**, select **Create templates for a new peripheral** as shown in [Figure 6-2](#), then click on **Next**.

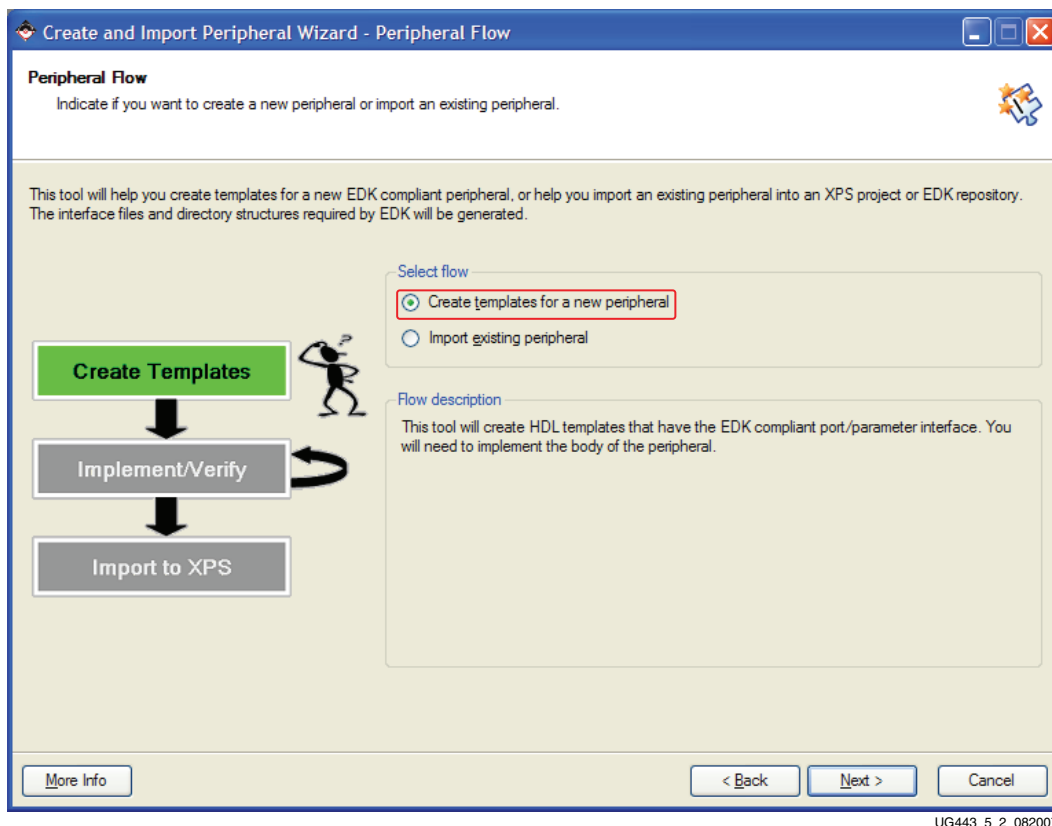


Figure 6-2: Create/IP Wizard Peripheral Flow

3. In the Create Peripheral - Repository or Project window, select a location for the repository where the cores will be generated as shown in Figure 6-3. then click on **Next**.

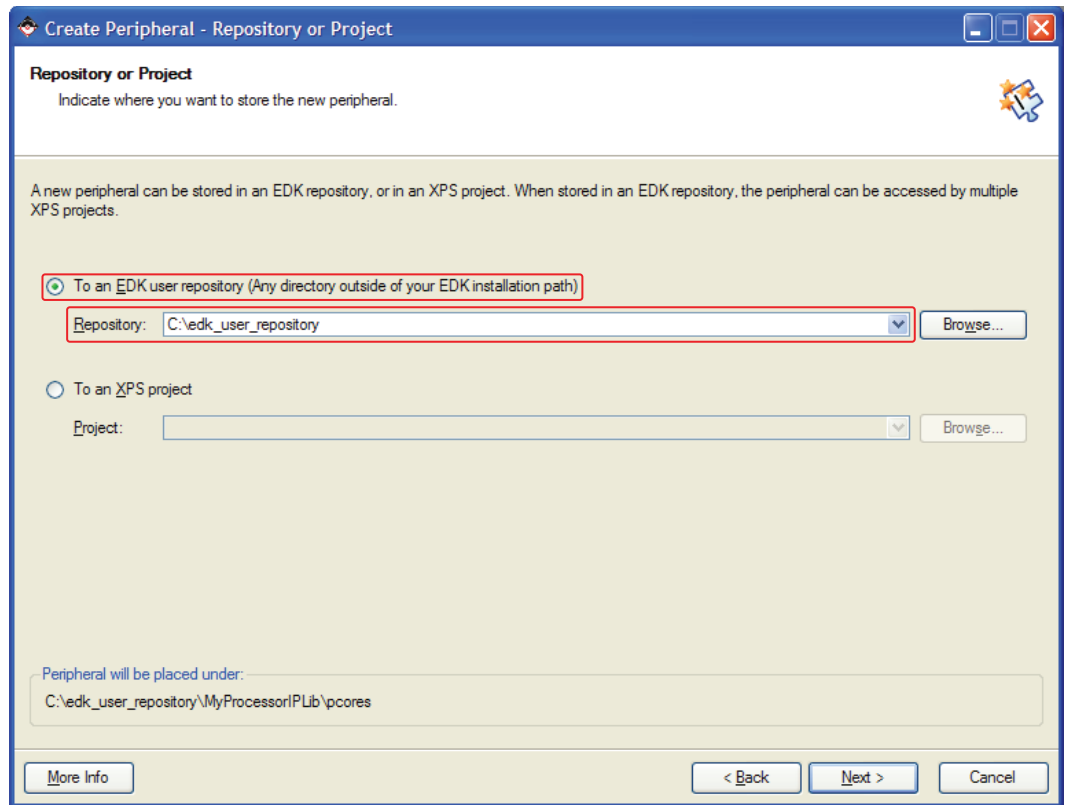
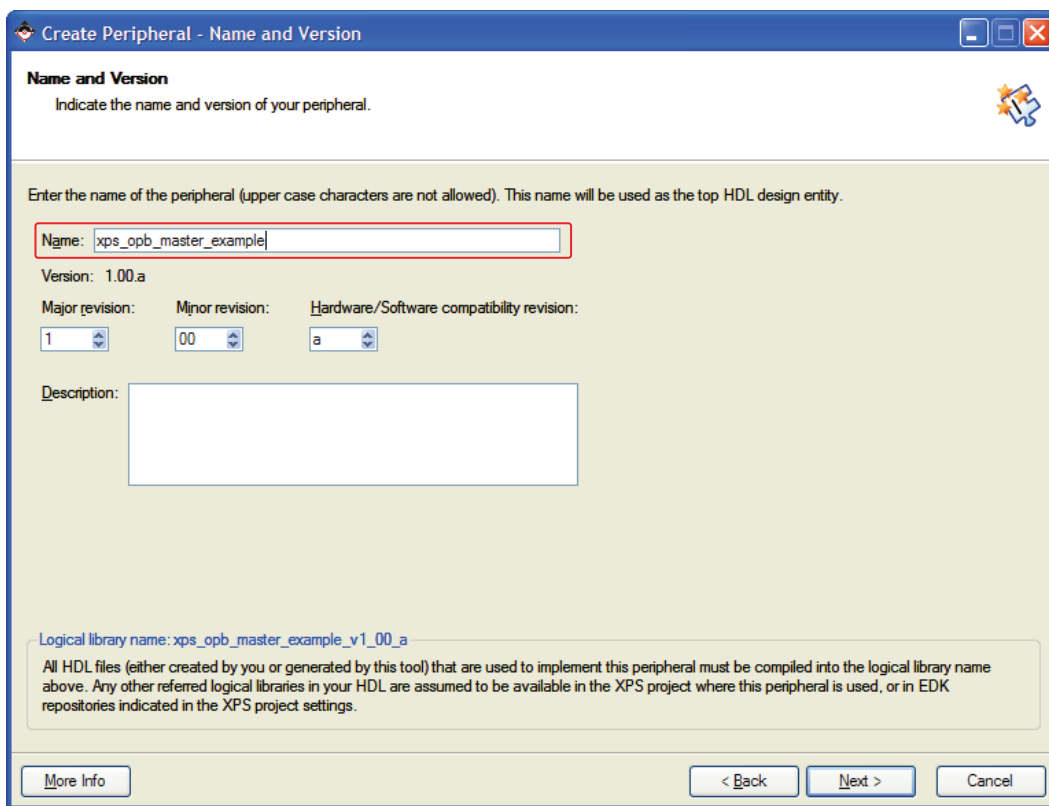


Figure 6-3: Create/IP Wizard Repository

4. In the Create Peripheral - Name and Version window, enter the name of the master/slave migrated core that will be created. The OPB migration example is shown in Figure 6-4. Click on **Next**.



Create Peripheral - Name and Version

Name and Version
Indicate the name and version of your peripheral.

Enter the name of the peripheral (upper case characters are not allowed). This name will be used as the top HDL design entity.

Name:

Version: 1.00.a

Major revision: Minor revision: Hardware/Software compatibility revision:

Description:

Logical library name: xps_opb_master_example_v1_00_a

All HDL files (either created by you or generated by this tool) that are used to implement this peripheral must be compiled into the logical library name above. Any other referred logical libraries in your HDL are assumed to be available in the XPS project where this peripheral is used, or in EDK repositories indicated in the XPS project settings.

UG443_5_4_082007

Figure 6-4: Create/IP Wizard Name and Version for OPB Migration

In the Create Peripheral - Name and Version window, enter the name of the master/slave migrated core that will be created. The PLB migration example is shown in [Figure 6-5](#). Click on **Next**.

Create Peripheral - Name and Version

Indicate the name and version of your peripheral.

Enter the name of the peripheral (upper case characters are not allowed). This name will be used as the top HDL design entity.

Name:

Version: 1.00 a

Major revision: Minor revision: Hardware/Software compatibility revision:

Description:

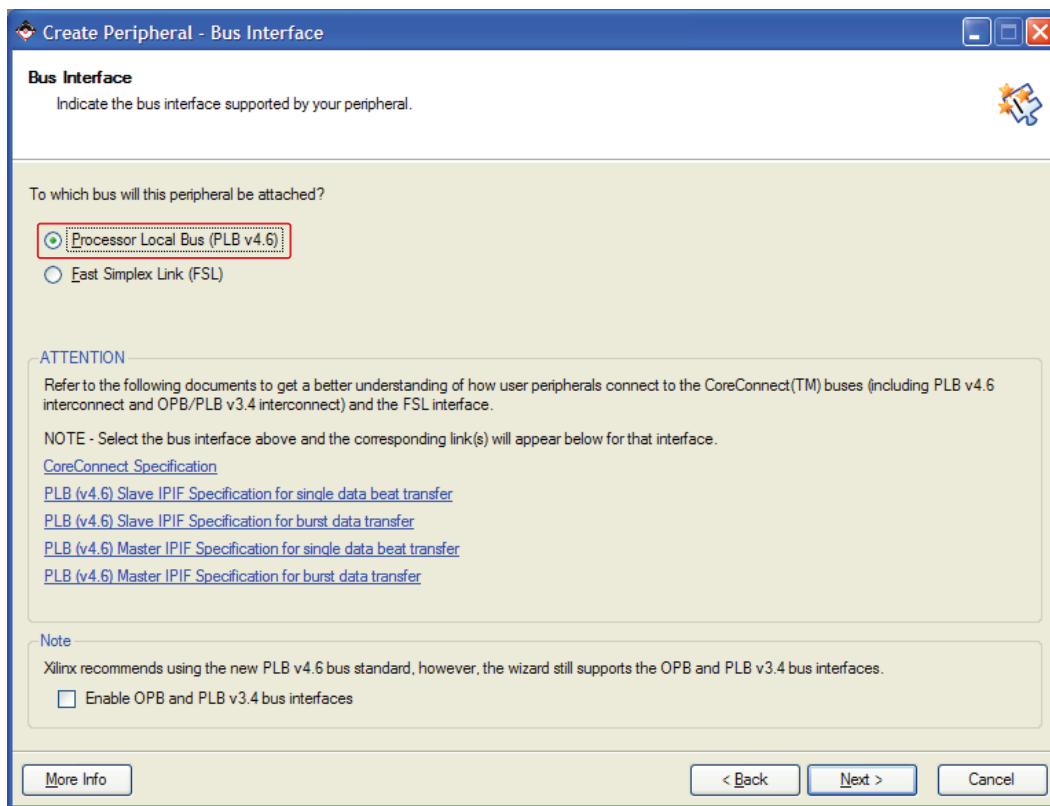
Logical library name: xps_plb_master_example_v1_00_a

All HDL files (either created by you or generated by this tool) that are used to implement this peripheral must be compiled into the logical library name above. Any other referred logical libraries in your HDL are assumed to be available in the XPS project where this peripheral is used, or in EDK repositories indicated in the XPS project settings.

UG443_5_5_082007

Figure 6-5: Create/IP Wizard Name and Version for PLB Migration

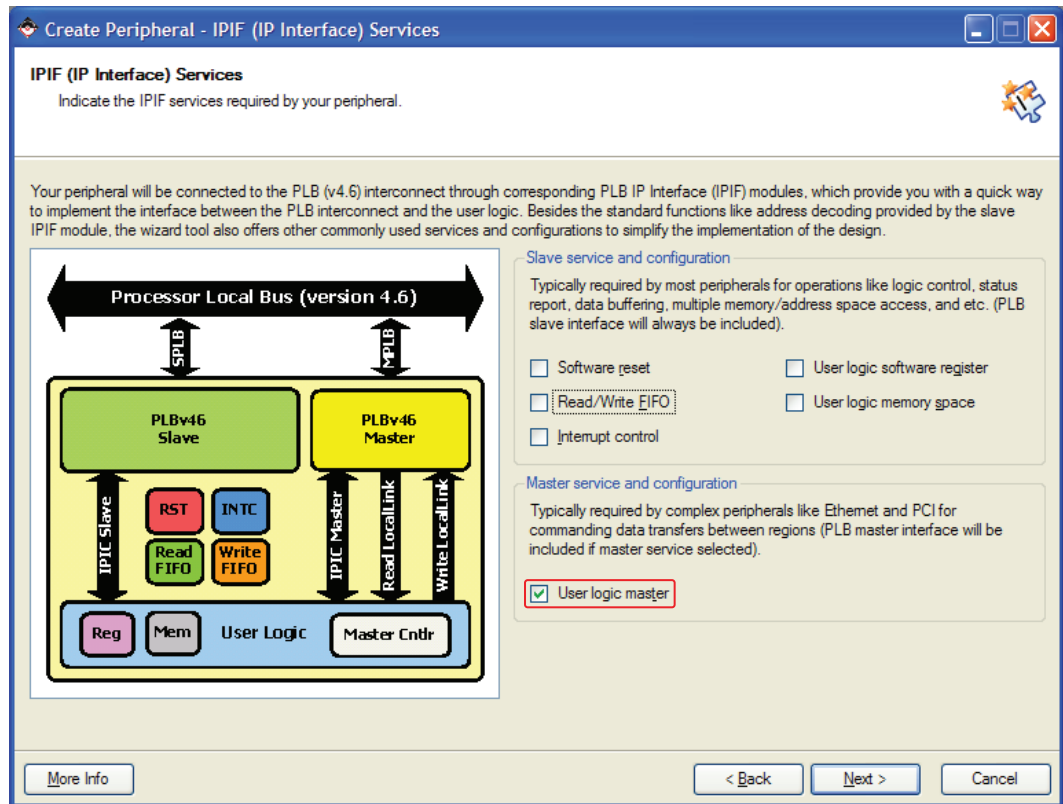
5. In the Create Peripheral - Bus Interface window, under **To which bus will this peripheral be attached?**, select **Processor Local Bus (PLB v4.6)** as shown in Figure 6-6, then click on **Next**.



UG443_5_6_082007

Figure 6-6: Create/IP Wizard Bus Interface

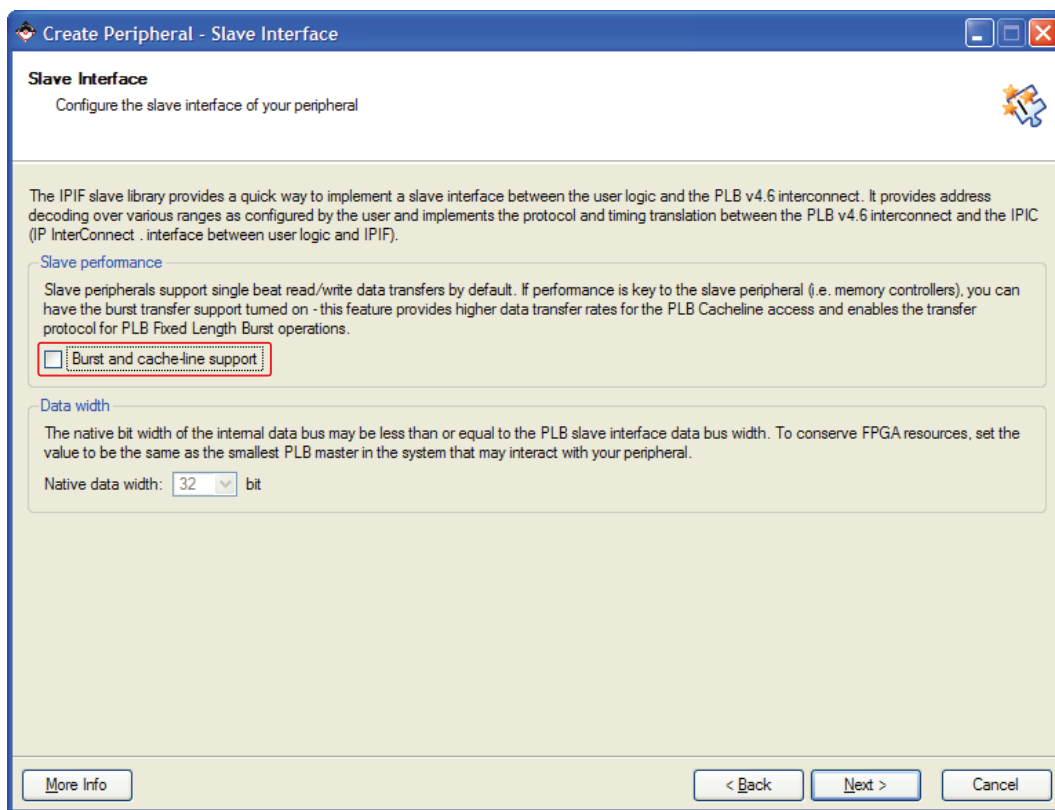
- In the Create Peripheral - IPIF (IP Interface) Services window, check **User logic master** as shown in [Figure 6-7](#). No slave services are selected. Click on **Next**.



UG443_5_7_082007

Figure 6-7: Create/IP Wizard IPIF Services

7. In the Create Peripheral - Slave Interface window, confirm that **Burst and cache-line support** is not selected as shown in [Figure 6-8](#), then click on **Next**.



UG443_5_8_082007

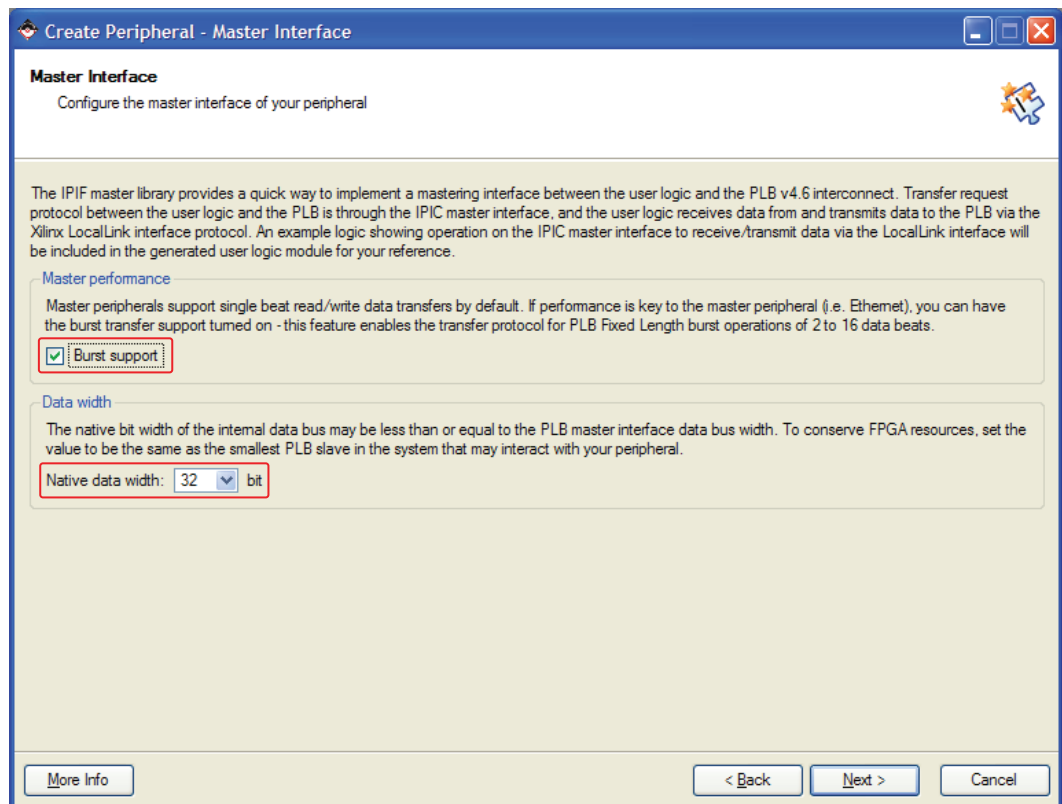
Figure 6-8: Create/IP Wizard Slave Interface

8. In the Create Peripheral - Master Interface window, several design considerations are needed.

Burst support should be checked if bursting support is needed or the native dwidth of the master needs to be **64** or **128**.

There are design considerations for the Native Dwidth if Burst support is enabled. If the Smallest Slave Native Dwidth < Master Native Dwidth, conversion cycles are required which takes additional resources. The native dwidth of the master should be equal to the smallest native dwidth of the slave inside the desired system to save resources.

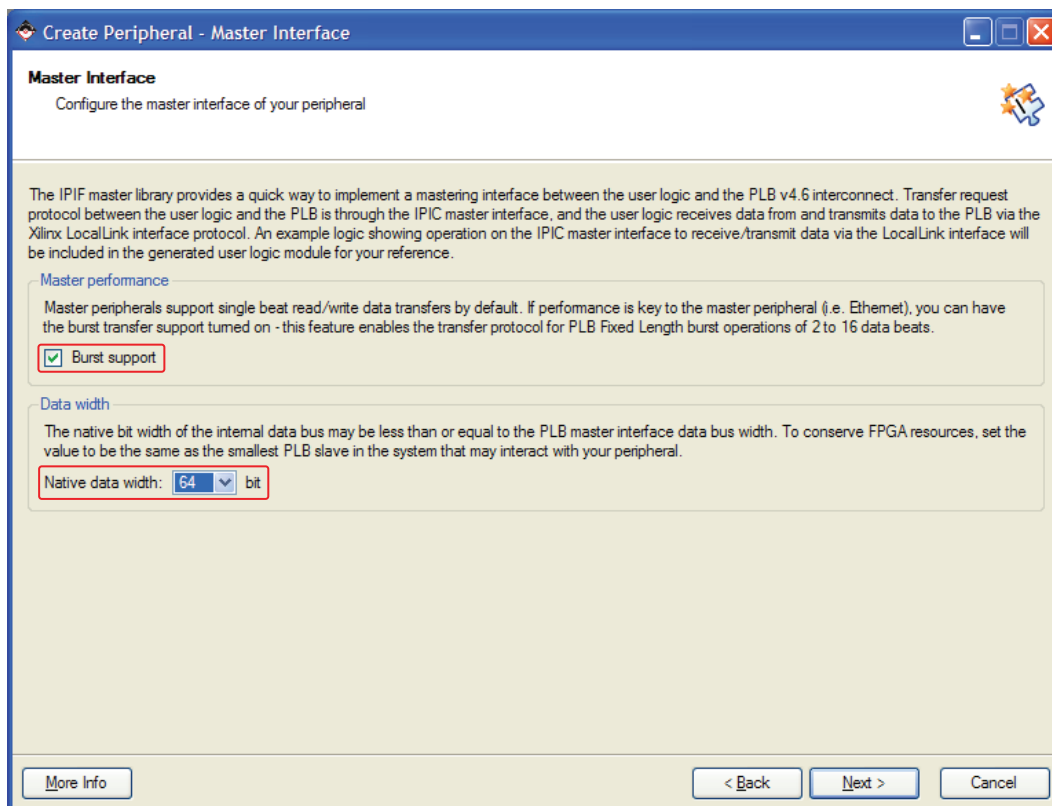
For the OPB example, **Burst support** is **enabled** and the **Native data width**: is set to **32** as shown in [Figure 6-9](#). Click on **Next**.



UG443_5_9_082007

Figure 6-9: Create/IP Wizard Master Interface for OPB Migration

For the PLB example, check **Burst support** and set the **Native data width** to **64** as shown in [Figure 6-10](#). Click on **Next**.



UG443_5_10_082007

Figure 6-10: Create/IP Wizard Master Interface for PLB Migration

- In the Create Peripheral - IP Interconnect (IPIC) window, make the selections shown in Figure 6-11. These are the standard IPIC signals connected to the user logic. Click on Next.

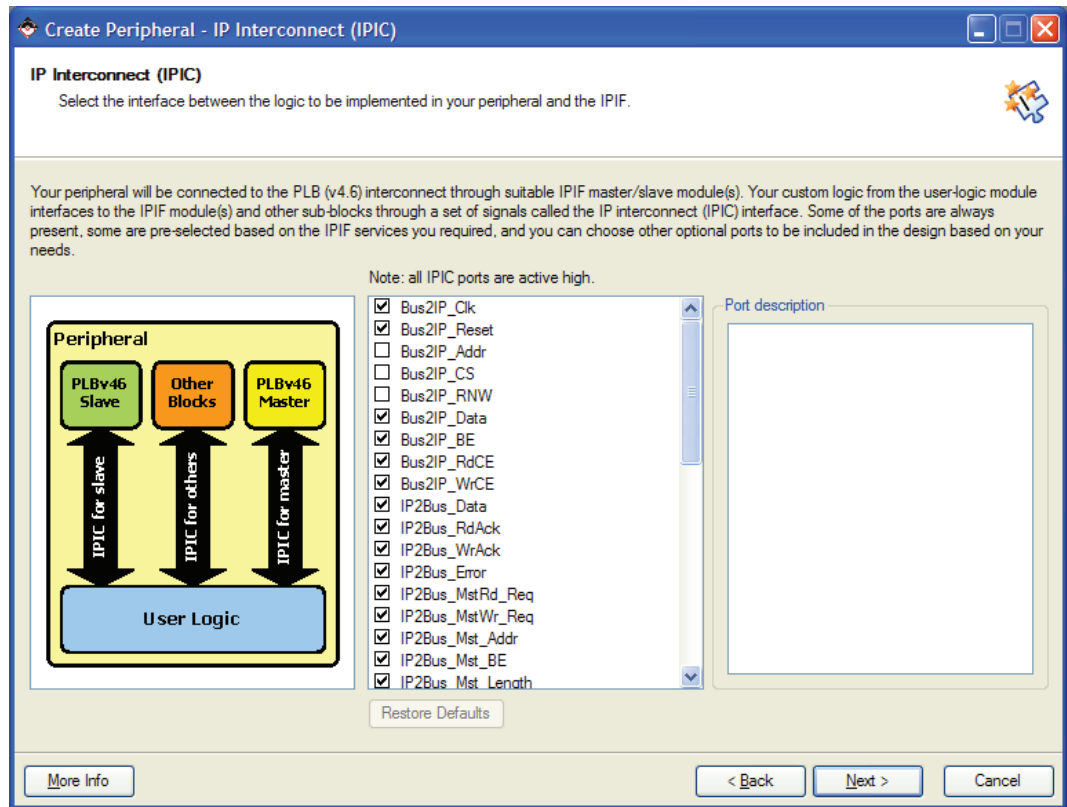


Figure 6-11: Create/IP Wizard IP Interconnect (IPIC)

10. In the (OPTIONAL) Peripheral Simulation Support window, the user can click on **Next** to continue or has the option of creating a BFM simulation environment to test the core by selecting **Generate BFM simulation platform for ModelSim-SE or ModelSim-PE**.

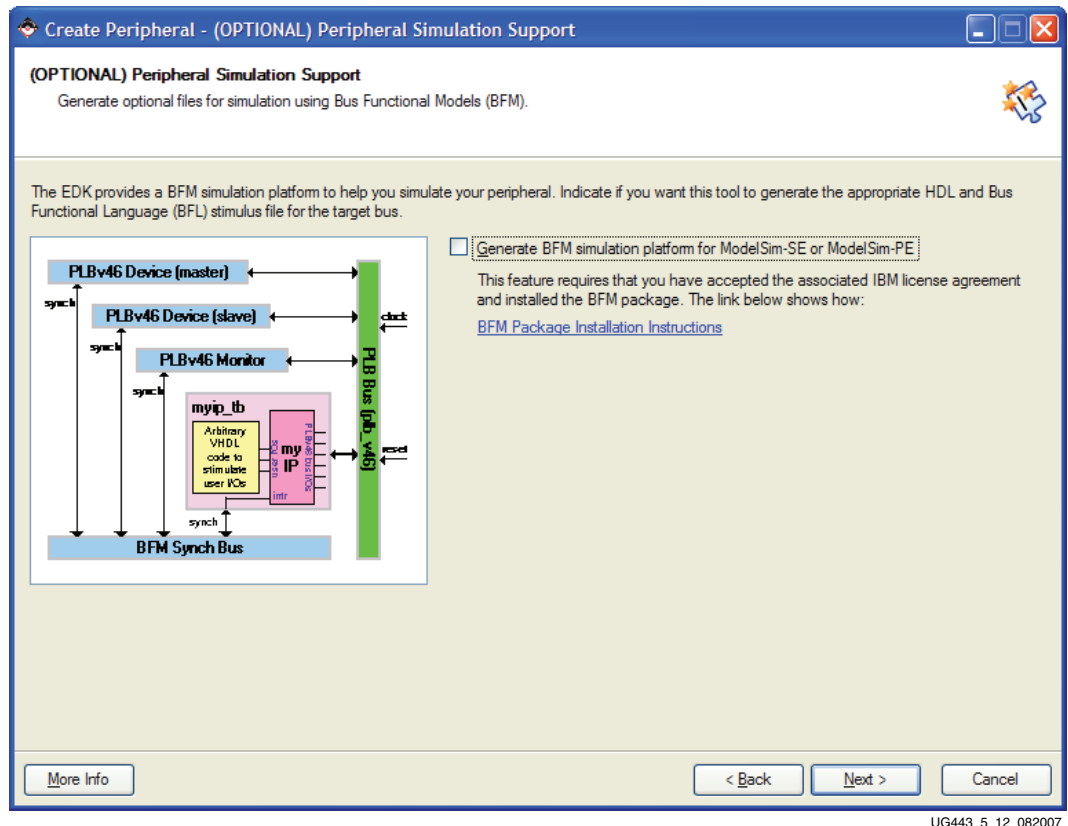
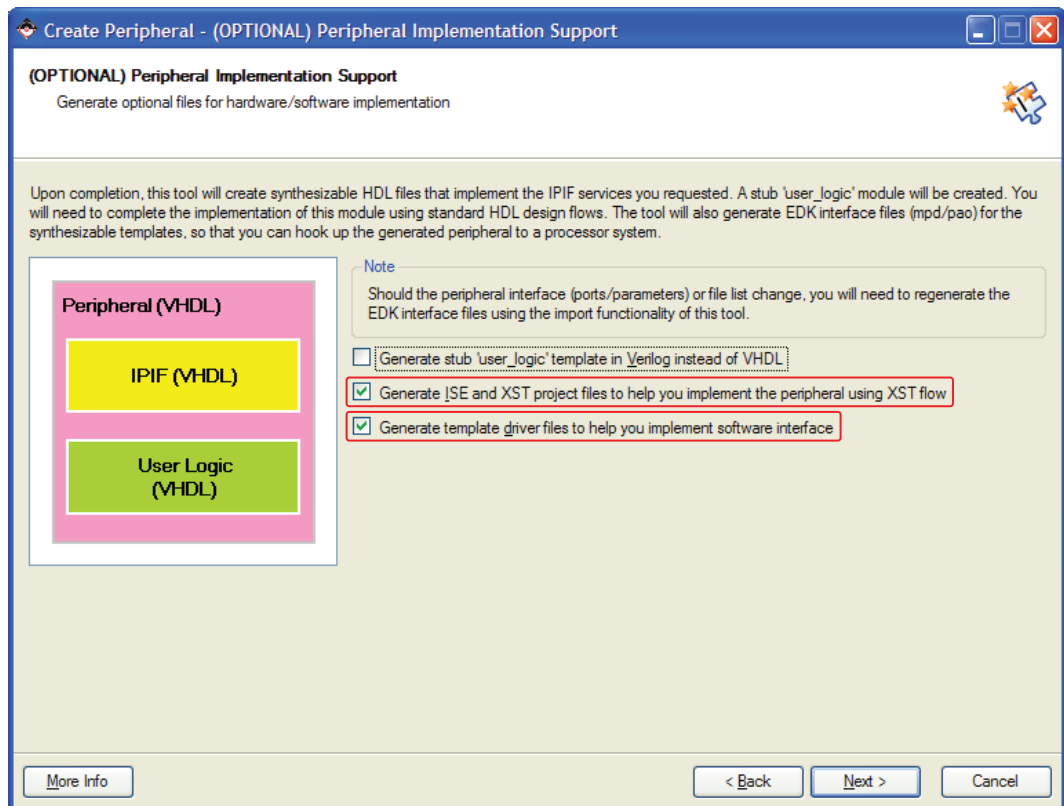


Figure 6-12: Create/IP Wizard Peripheral Simulation Support

11. In the Create Peripheral - (OPTIONAL) Peripheral Implementation Support window, make the selections as shown in [Figure 6-13](#):
 - a. Check **Generate ISE and XST project files to help you implement the peripheral using XST flow**
 - b. Check **Generate template driver files to help you implement software interface**
 - c. Click on **Next**.



UG443_5_13_082007

Figure 6-13: Create/IP Wizard (OPTIONAL) Peripheral Implementation Support

12. In the Create Peripheral - Finish window, click on **Finish** to generate the core as shown in Figure 6-14.

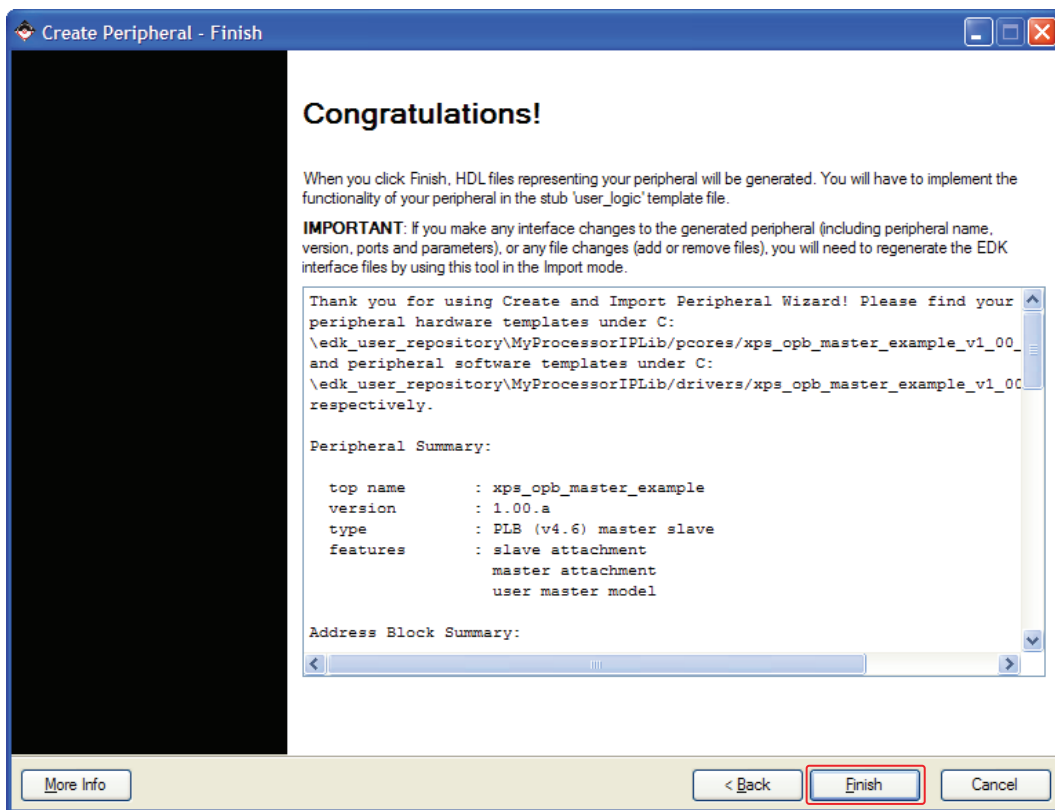


Figure 6-14: Create/IP Wizard Finish

Parameters for PLBV46 Master/Slave Cores

Certain parameters for the PLBV46 Master interface are necessary for the tools. [Table 6-1](#) shows these parameters. The necessary parameters for the PLBV46 Slave interface is shown in the previous chapter.

Note: Some of the parameters may be overwritten inside EDK to system values depending on the system.

Table 6-1: PLBV46 Master Necessary Parameters

Parameter	Description	Allowable Values
C_MPLB_NATIVE_DWIDTH	Specifies the internal native data width of the master.	32,64,128
C_MPLB_AWIDTH	Width of the PLB Address Bus.	32
C_MPLB_DWIDTH	Width of the PLB Data Bus.	32,64,128
C_MPLB_SMALLEST_SLAVE	Indicated the smallest Native Dwidth of any Slave attached to the PLBV46 used by the master.	32,64,128

The PLBV46 Master Burst parameters are described in [Table 6-2](#). The C_INHIBIT_CC_BLE_INCLUSION parameter is set to 0 to automatically include the logic for conversion cycles and burst length expansion.

Table 6-2: PLBV46 Master Burst Parameters

Parameter	Description	Allowable Values
C_MPLB_NATIVE_DWIDTH	Specifies the internal native data width of the master.	32,64,128
C_MPLB_AWIDTH	Width of the PLB Address Bus.	32
C_MPLB_DWIDTH	Width of the PLB Data Bus.	32,64,128
C_MPLB_SMALLEST_SLAVE	Indicated the smallest Native Dwidth of any Slave attached to the PLBV46 used by the master.	32,64,128
C_INHIBIT_CC_BLE_INCLUSION	Parameter is used to override the automatic inclusion of the Conversion Cycle and Burst length Expansion logic.	0,1

The PLBV46 Master Single parameters are described in [Table 6-3](#).

Table 6-3: PLBV46 Master Single Parameters

Parameter	Description	Allowable Values
C_MPLB_NATIVE_DWIDTH	Specifies the internal native data width of the master.	32
C_MPLB_AWIDTH	Width of the PLB Address Bus.	32
C_MPLB_DWIDTH	Width of the PLB Data Bus.	32,64,128

