XAPP935 (v1.1) June 7, 2007

Reference System: PLB DDR2 with OPB Central DMA

Author: James Lucero

Abstract

This reference system demonstrates the functionality of the Processor Local Bus (PLB) Double Data Rate 2 (DDR2) Synchronous DRAM (SDRAM) memory controller in a PowerPC® 405 (PPC) 405 processor system. The On-Chip Peripheral Bus (OPB) Central DMA controller is also included in this system to test DMA operations on the memory controller.

The DDR2 memory space is split into a cacheable region of memory and a non-cacheable region of memory. The OPB Central DMA stand-alone software application provided with this reference system is executed out of the cacheable region of memory and the DMA transfers occur inside the non-cacheable region of memory. The Error Correcting Code (ECC) stand-alone software application tests the functionality of the PLB DDR2 memory controller with ECC.

This application note describes how to set up parameters for the PLB DDR2 memory controller with ECC, the OPB Central DMA controller, and the OPB2PLB bridge for communication between the OPB and PLB bus. This reference system is targeted for the Xilinx ML410 Rev B board.

Included Systems

Included with this application note is one reference system (ml410_ppc_plb_ddr2) built for the Xilinx ML410 Rev B board. The reference system is available for download at:

- www.xilinx.com/bvdocs/appnotes/xapp935.zip

Introduction

There are many advantages to DDR2 compared to DDR. For example, with DDR2, the memory can achieve higher clock speeds. In addition, DDR2 includes On Die Termination (ODT) to improve signal integrity. Lastly, differential data strobes provide improved memory timing. The PLB DDR2 memory controller from Xilinx supports these features through parameters.

Hardware and Software Requirements

The hardware and software requirements are:

- Xilinx ML410 Rev B board
- Xilinx Platform USB cable or Parallel IV programming cable
- RS232 serial cable and serial communication utility (HyperTerminal)
- Xilinx Platform Studio 9.1.02i
- Xilinx Integrated Software Environment (ISE™) 9.1.03i

Reference System Specifics

This system uses the PPC 405 processor with a processor and bus clock frequency of 100 MHz. The reference system includes PLB DDR2, OPB UART Lite, OPB INTC, OPB GPIO and 32KB of PLB BRAM. The parameters and settings for PLB DDR2, OPB Central DMA, and the OPB2PLB bridge are described in the next section of this application note. Refer to Figure 1 for the block diagram and Table 1 for the address map of the system.

Block Diagram

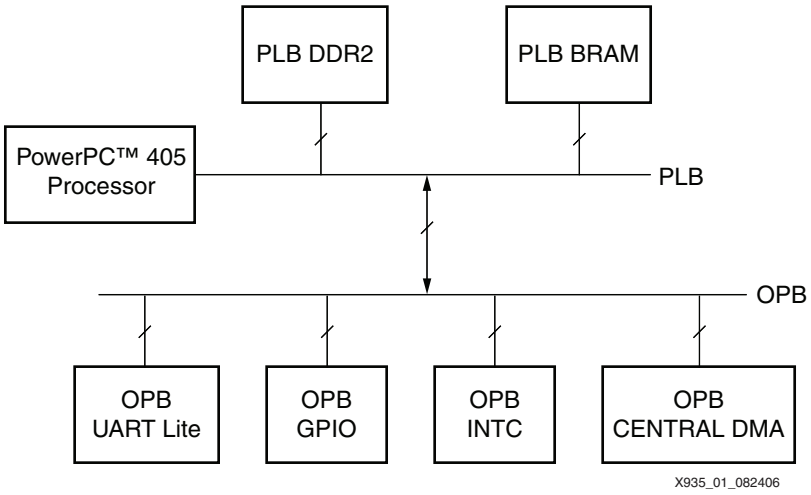

Figure 1: **Reference System Block Diagram**

Address Map

Table 1: **Reference System Address Map**

Peripheral	Instance	Base Address	High Address
plb_bram	plb_bram_if_cntlr_0	0xFFFF8000	0xFFFFFFFF
plb_ddr2	plb_ddr2_0	0x00000000	0x0FFFFFFF
opb_uartlite	RS232_Uart_1	0x40600000	0x4060FFFF
opb_gpio	LEDs_8Bit	0x40000000	0x4000FFFF
opb_intc	opb_intc_0	0x41200000	0x4120FFFF
opb_central_dma	opb_central_dma_0	0x41E00000	0x41E0FFFF

Modifications Required to the Initial System

The OPB Central DMA controller is used to provide data transfer inside the non-cacheable region of PLB DDR2 memory space. In order for the OPB DMA controller to communicate to the PLB memory, the OPB2PLB bridge is included.

The following sections explain the configuration of the PLB DDR2 memory controller with ECC, the OPB Central DMA controller, and the OPB2PLB bridge.

Setting PLB DDR2 Parameters

The PLB DDR2 memory controller supports differential data strobes. By selecting the **Enable Differential Signaling for DDR2 Memory** inside the **User/DDR2 Parameters** tab**,** the core will drive the DQS and DQSn as two unique IO buffers. Therefore, the DQSn signal does not require a differential IO buffer. This parameter is not set inside this reference system, so the DQSn signal is not connected.

The following parameters for the PLB DDR2 memory controller are under the **User/Misc** tab of the PLB DDR2 core as shown in Figure 2. By selecting **Support PLB Burst and Cacheline Transactions**, this parameter enables caching for the software application. In addition, **Support Registered DIMM** is enabled since the board's DDR2 DIMM supports this feature.

The **DDR2 Clock Period** is set to 5000ps (200 MHz). For the PLB DDR2 memory controller's clocking structure and supported clock configurations, refer to the DDR2 SDRAM Clocking

section of the *PLB Double Data Rate (DDR2) Synchronous DRAM (SDRAM) Controller* Product Specification.

Number of IDELAYCTRL Primitives (V4 only) that are explicitly instantiated is set to **4**. The constraints for the IDELAY controller are placed inside the **LOC Constraints of IDELAYCTRL Primitive**. For more detailed information on constraining the IDELAY controllers, refer to the IDELAY Constraints section inside the *PLB Double Data Rate (DDR2) Synchronous DRAM (SDRAM) Controller* Product Specification.

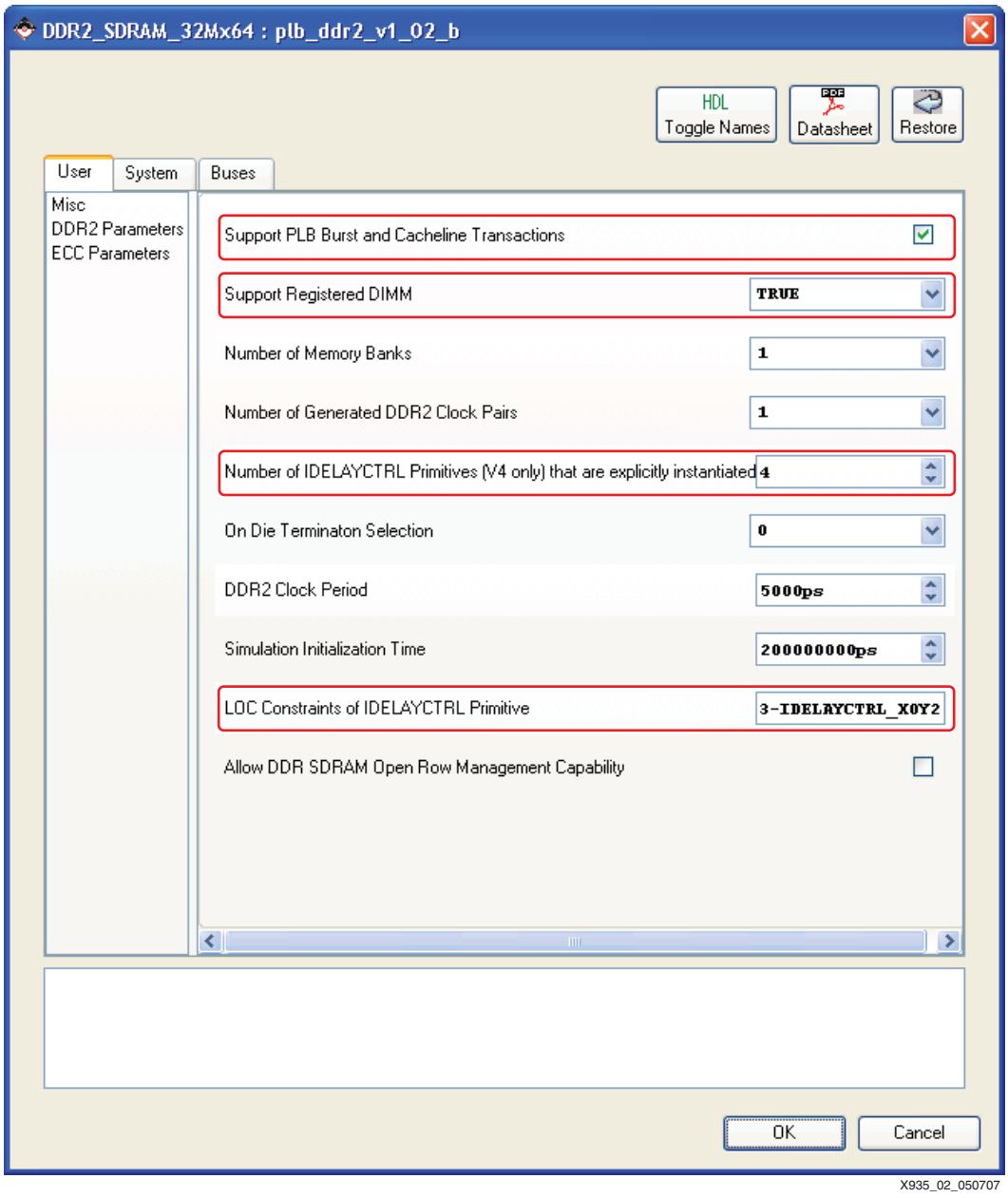

Figure 2: **Setting the PLB DDR2 User/Misc Parameters**

Adding ECC to The PLB DDR2 Memory Controller

Since the DDR2 DIMM supports 64-bits and the memory controller is using 32-bits for data, ECC functionality can be added to the PLB DDR2 memory controller. With ECC enabled, two

separate memory addresses exists for the actual DDR2 SDRAM's memory and for the ECC registers.

Special consideration is needed when running code from the DDR2 SDRAM's memory space when ECC is enabled. Inside this reference system, a program that clears DDR2 memory is initialized inside BRAM when downloading the bitstream. By writing zeros to the memory, the ECC logic will create check bits for all memory addresses defined inside the program. This prevents the possibility of getting false ECC errors by having uninitialized check bits in addresses where the program code is executed. If the memory is not cleared and the check bits remain uninitialized, false ECC errors can occur when performing cachline reads of the program that may access data beyond what was written when the program was downloaded.

To add ECC functionality; parameters for the PLB DDR2 memory controller must be modified, ECC ports must be added and connected, and the ECC address range for the ECC registers must be defined in the system.

1. Adding parameters for ECC inside PLB DDR2

Right click on **PLB DDR2** and select **Configure IP**. Under **User/ECC Parameters**, set **Include Support for ECC Logic**, **Enable Use of ECC Registers** and, **Support ECC Force Error Condition** to **1** as shown in Figure 3. These parameters allow for ECC functionality to be supported and the ECC functionality tested in the ECC software application.

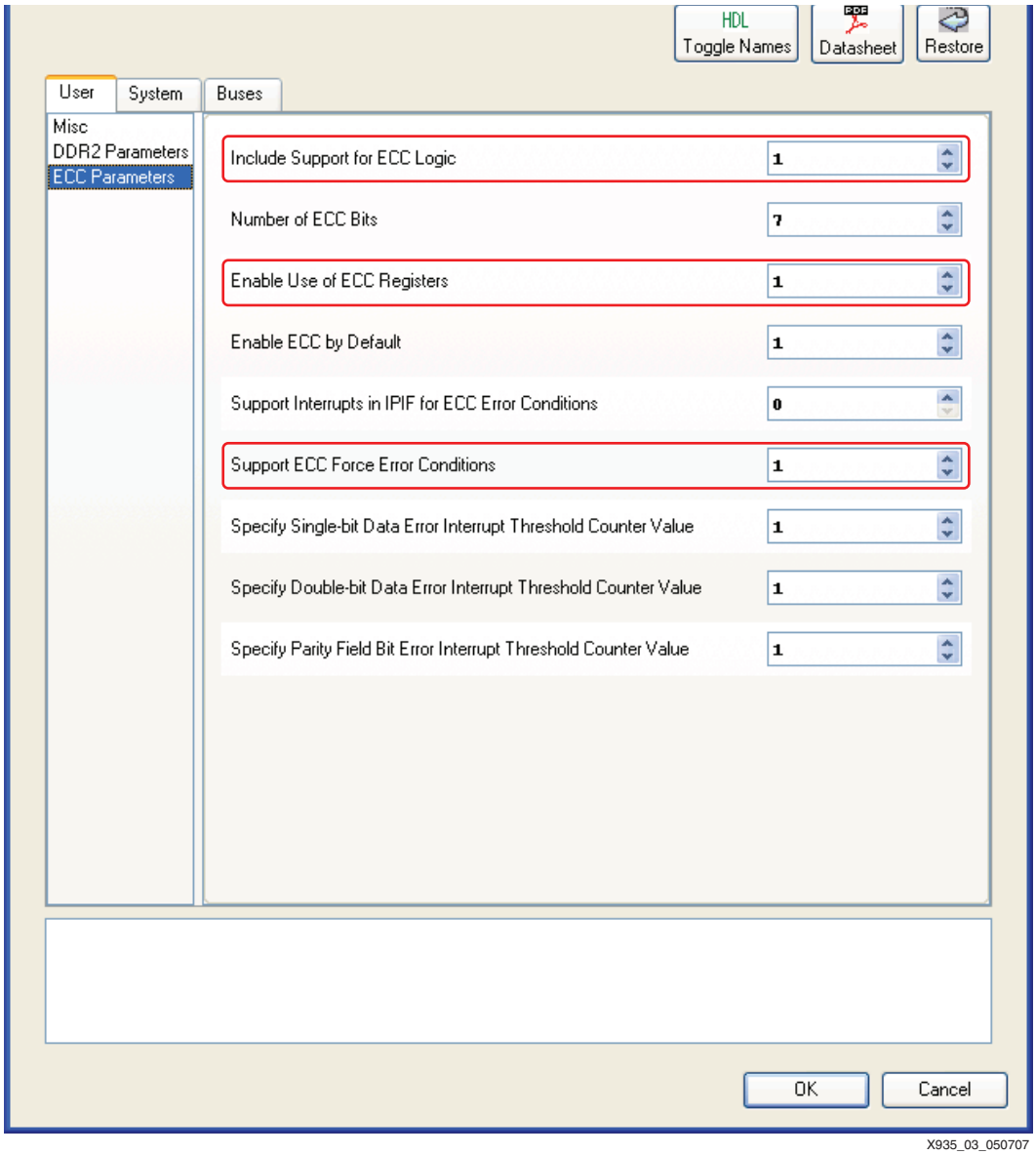

Figure 3: **Setting the PLB DDR2 User/ECC Parameters**

2. Adding External Ports for ECC

Inside the **Ports Filter**, expand the tree node for PLB DDR2. Make ports **DDR_DM_ECC**, **DDR_DQ_ECC**, **DDR_DQS_ECC** external by setting the **Net** name to **Make External**. Note that an external net name is entered as shown in Figure 4. These external pins must be connected in the UCF based upon the board's pin constraints.

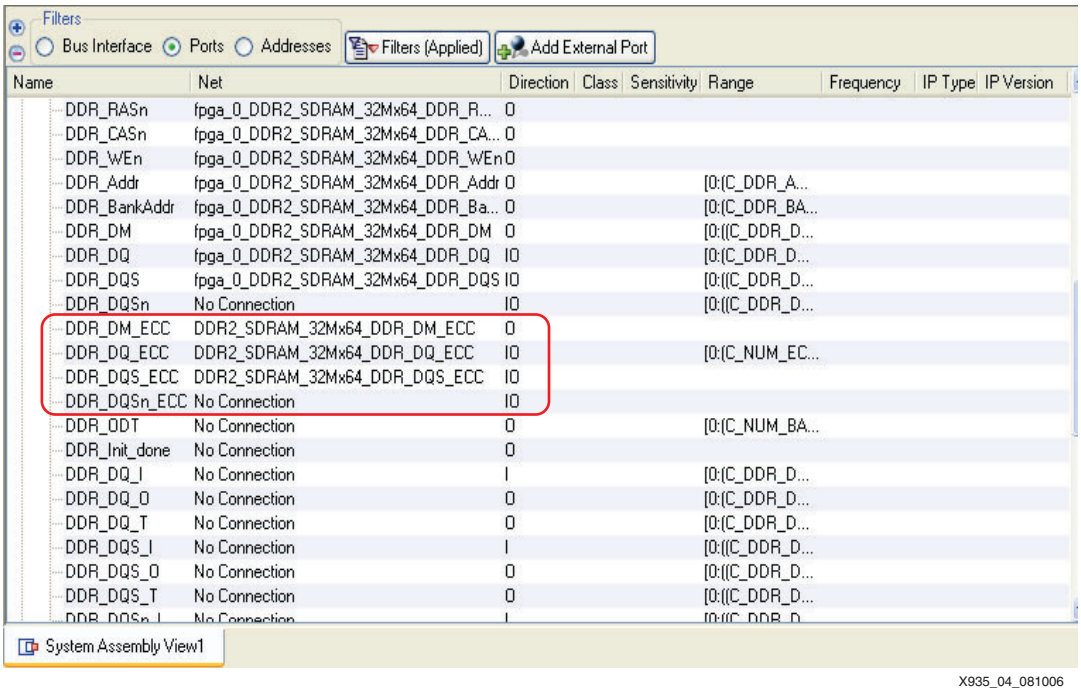

Figure 4: **Ports Filter, Adding ECC Ports**

3. Setting ECC Address Range

Go to **Addresses Filter** and set the base address for **PLB DDR2 ECC** to `0x30000000` with a **Size** of **64KB**. This address range is for the ECC registers.

Adding the OPB Central DMA Controller

Four steps are needed to add the OPB Central DMA controller to the system. The first step connects OPB Central DMA to the system. The second step sets the parameters for the OPB Central DMA core. The third step connects the interrupt port for the OPB Central DMA controller. The fourth step assigns the address for the OPB Central DMA controller.

1. Connecting OPB Central DMA To The System

Under the **Bus Interface Filter**, expand the **opb_central_dma_0** tree node and connect **opb_central_dma_0 sopb** as slave and **opb_central_dma_0 mopb** as master by clicking on slave and master, respectively, both in the tree underneath the OPB bus as shown in Figure 5.

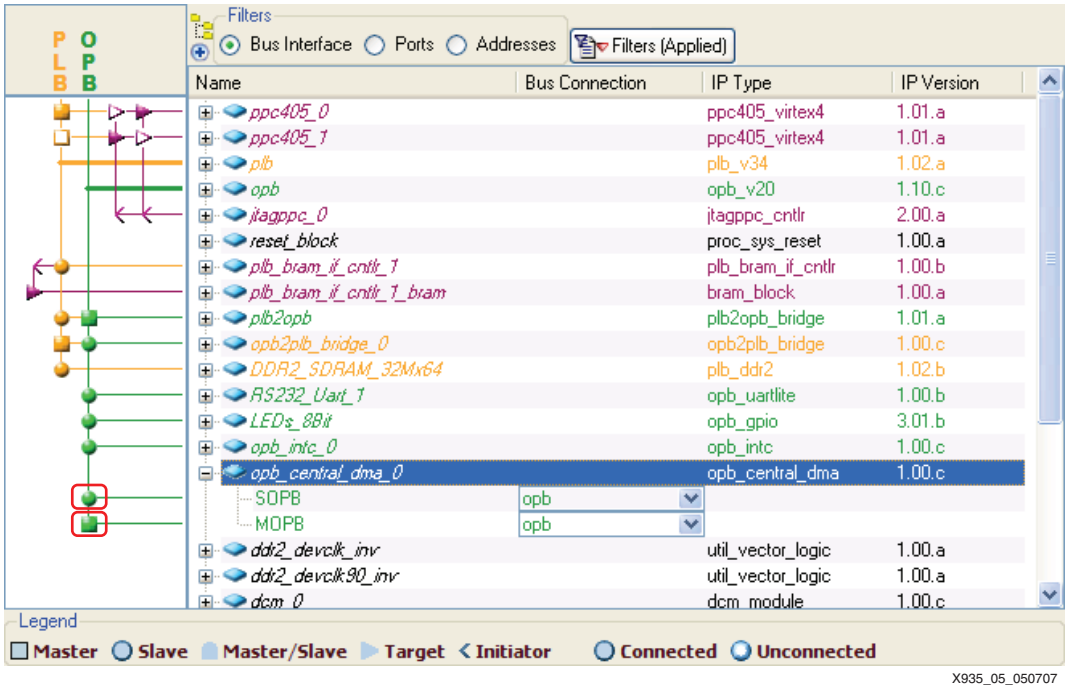

Figure 5: **Bus Interface Filter, Setting Master and Slave for OPB Central DMA**

2. Right click on **OPB Central DMA** and select **Configure IP**. Under **User\All** tab set the **C_READ_OPTIONAL_REGS** to **1** as shown in Figure 6. This enables the option to read OPB Central DMA registers, allowing the OPB Central DMA software application to check if the DMA interrupt is properly cleared after the DMA operation is completed.

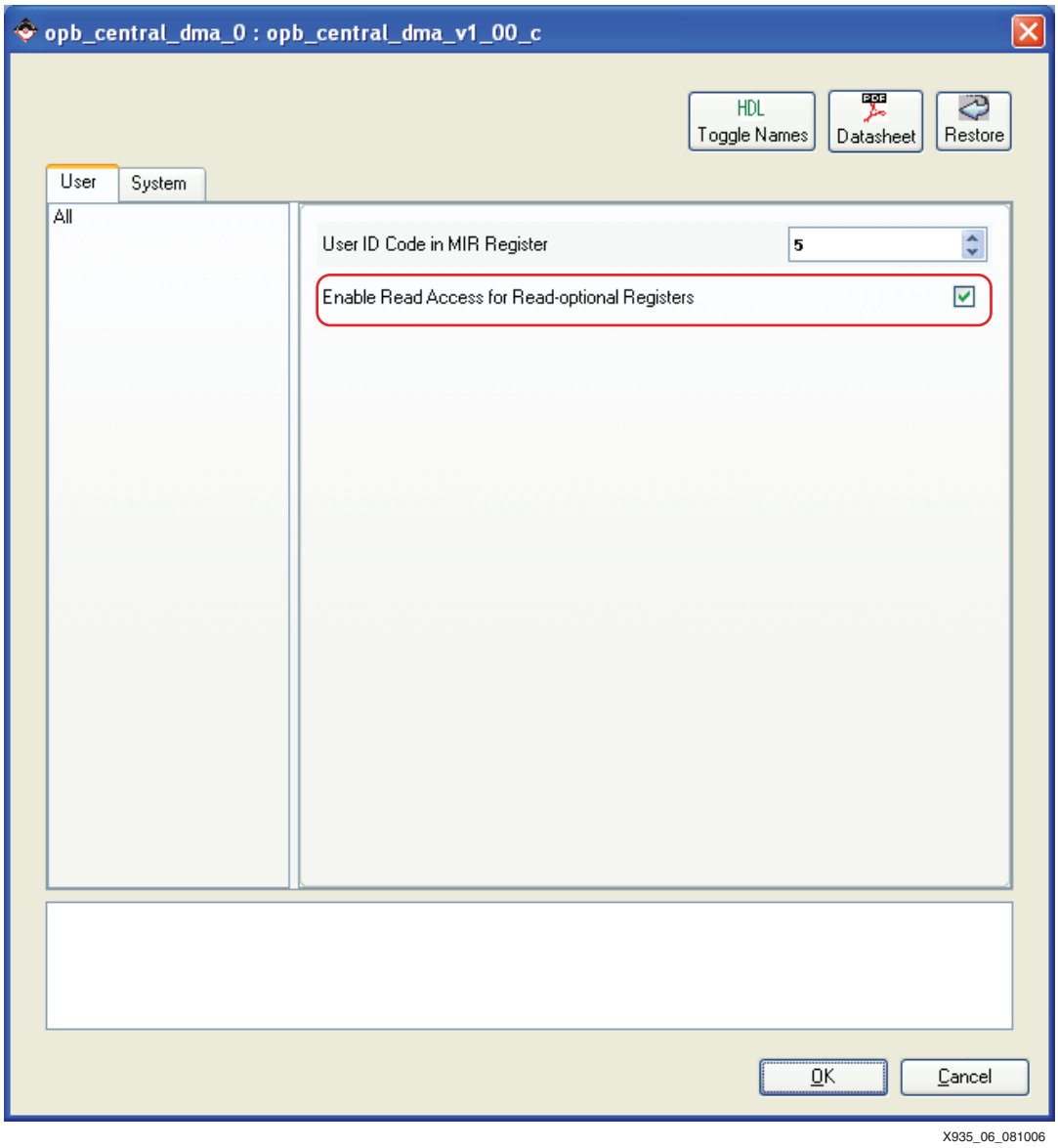

Figure 6: **Setting OPB Central DMA User/All Parameters**

3. Connecting the Interrupt Port for OPB Central DMA

Choose the **Ports Filter** and expand the opb_central_dma_0 tree node. Connect the output port **DMA_Interrupt** to **DMA_Irpt**. Obtain the list of interrupts by expanding the opb_intc_0 tree node and clicking on the last port under the **Net** for **Intr**. This will bring up the **Interrupt Connection Dialog** box. Add the **DMA_Irpt** output to Connected Interrupts as shown in Figure 7.

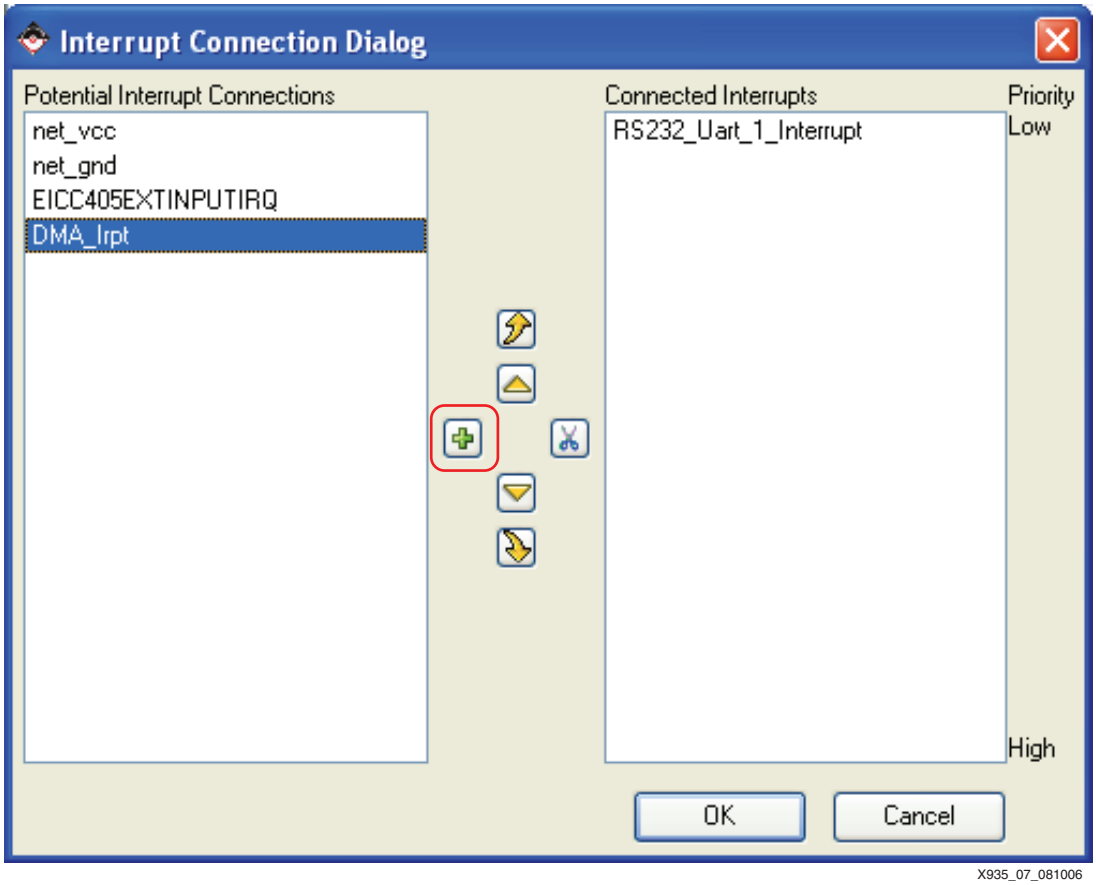

Figure 7: **Interrupt Connect Dialog Box, Adding OPB Central DMA Interrupt**

4. Setting OPB Central DMA Address Range

In **Address Filter View**, select the base address for **opb_central_dma_0** of `0x41E00000` and a **Size** of **64KB**.

Adding the OPB2PLB Bridge

In order for OPB Central DMA operations to be performed on the PLB DDR2 memory controller on the PLB bus, a bridge must be connected between the OPB and PLB buses. Three steps must be completed in order for the bus to be added to the system. The first step configures the bridge's parameters and connects the bus interface. The second step connects the port connections between buses. Finally, the PLB address range for the bridge is specified.

1. Adding the OPB2PLB Bridge to the System

In the **Bus Interface Filter**, connect the opb2plb_bridge_0 **MPLB** to **PLB** and **SOPB** to **OPB** as seen in Figure 8.

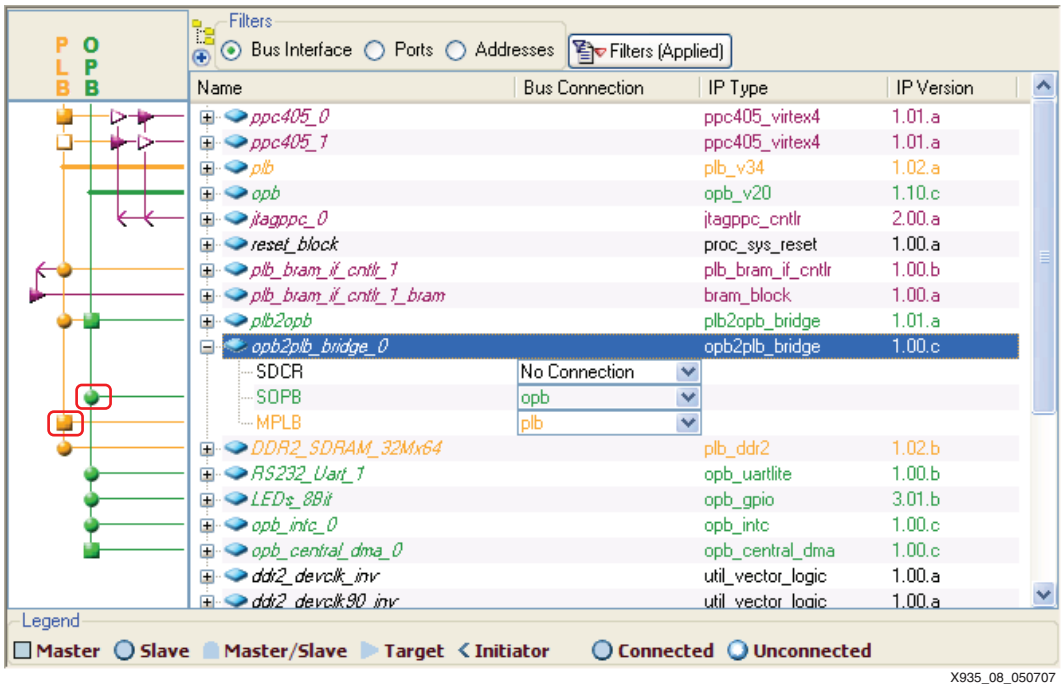

Figure 8: **Bus Interface Filter, Setting Master and Slave for OPB2PLB Bridge**

Right click on opb2plb_bridge_0 and select **Configure IP**. Set the **User\Bridge** Features parameter **OPB and PLB Clocks are Same Frequency** to **1**.

2. Connecting the OPB2PLB Bridge Ports

Go to the **Ports Filter** and expand the plb2opb tree node. Connect the output port **BGI_Trans_Abort** to **bgi_trans_abort**. Expand opb2plb_bridge_0 tree node and connect **BGI_Trans_Abort** input port to **bgi_trans_abort**.

3. Setting Address Ranges for the OPB2PLB Bridge

In the **Bus Interface Filter**, select the base address range for **opb2plb_bridge_0 RNG0** to `0x000000000` with a **Size** of **1G**. This covers the address range for all PLB devices.

The Software Application

OPB Central DMA Software Application

The first software application is executed from the cacheable region of main memory, while DMA operations are occurring in the non-cacheable region of main memory. The first 128 MB of memory are cacheable while the other 128 MB of memory are non-cacheable. The software application's source code determines the location and size of the cacheable memory inside DDR2 through caching functions for the PPC 405.

At the start of the software application, the memory regions assigned to the source address and the destination address are cleared. Then data is written to the memory block of the source address. The OPB Central DMA core is initialized and then set up to use interrupts. DMA operations start when the source base address and destination base address are written to the appropriate OPB Central DMA register. When the transfer is complete, an interrupt occurs.

When DMA operations are complete, data at the source addresses are compared with data from the destination addresses to ensure the data was transferred correctly. Also, the software application clears the interrupt caused by the DMA operation.

The software application described above is executed out of DDR2 memory. The software application is found under the project root directory `TestApp_MemoryCaching/src/TestApp_MemoryCaching.c`.

ECC Software Application

The second software application demonstrates ECC functionality for the PLB DDR2 core. The two tests (single bit and double bit errors) start by resetting the ECC Control Register (ECCCR), ECC Status Register (ECCSR), ECC Single Count Register (ECCSEC), and ECC Double Count Register (ECCDEC) to default values.

The first test in the software application creates single bit errors by asserting the Force Single Bit Error bit inside the ECCCR register. Then a memory location aligned on a 64-bit wide vector is written and read in the DDR2 memory. Since single bit errors are automatically corrected, the software application verifies that the data read from the memory location is the same data as the data written to the memory location. The PLB DDR2 controller updates the ECCSR register with the appropriate bits being set and increments the ECCSEC register to indicate the detection of single bit errors. Both of these registers values are verified by the software application.

The second test in the software application creates double bit errors by asserting the Force Double Bit Error bit inside the ECCCR register. Then a memory location aligned on a 64-bit wide vector is written and read in the DDR2 memory. Since double bit errors can not be corrected, the software application verifies that the data read from the memory location is **not** the same as the data that was written to the memory location. The PLB DDR2 controller updates the ECCSR register with the appropriate bits being set and increments the ECCDEC register to indicate the detection of double bit errors. Both of these registers values are verified by the software application.

After the software application is complete, the ECCCR, ECCSR, ECCSEC, and ECCDEC registers are reset to default values.

The above software application is run out of PLB BRAM. The software application is found under the project root directory `TestApp_MemoryECC/src/TestApp_MemoryECC.c`.

Executing the Reference System

Using HyperTerminal or a similar serial communications utility, map the utility's operation to the physical COM port to be used. Then connect the board's UART to this COM port. Set the HyperTerminal to the Bits per second of **9600**, Data Bits to **8**, Parity to **None,** and Flow Control to **None**. See Figure 9 for the settings. This is necessary to see the results from the software application.

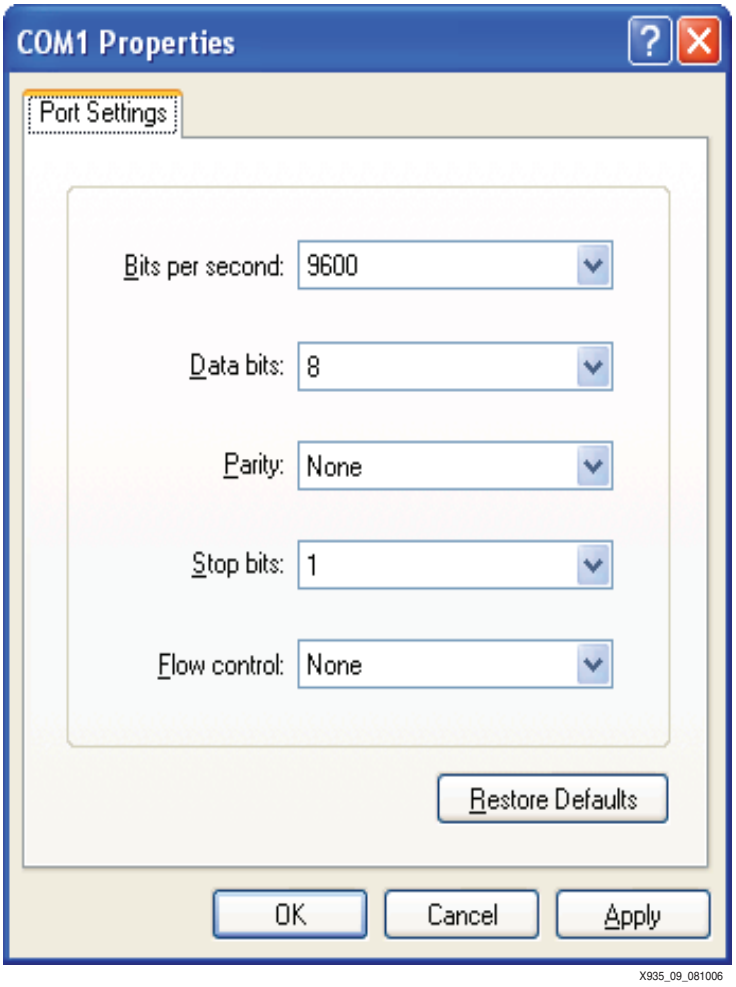

Figure 9: **HyperTerminal Settings**

Executing the Reference System using the Pre-Built Bitstream and the Compiled Software Applications

To execute the system using files inside the `ready_for_download/` in the project root directory, follow these steps:

1. Change directories to the `ready_for_download` directory.
2. Use iMPACT to download the bitstream by using the following:

   `impact -batch xapp935.cmd`
3. Invoke XMD and connect to the PowerPC 405 processor by the following command:

   `xmd -opt xapp935.opt`
4. Download the executables by the following command:

   `dow TestApp_MemoryCaching.elf` or **`dow TestApp_MemoryECC.elf`**

Executing the Reference System from EDK

To execute the system using EDK, follow these steps:

1. Open `system.xmp` inside EDK.
2. Use **Hardware→Generate Bitstream** to generate a bitstream for the system.

3. Use **Software→Build All User Applications** to build the software applications.
4. Download the bitstream to the board with **Device Configuration→Download Bitstream**.
5. Launch XMD with **Debug→Launch XMD...**
6. Download the executables by the following command:

   `dow TestApp_MemoryCaching.elf` or **`dow TestApp_MemoryECC.elf`**

Running the Software Applications

To run the either of software applications, use the ***`run`*** command inside XMD. The status of the software application is displayed in the HyperTerminal data screen.

Running the OPB Central DMA Software Application

Once the DMA operations are complete and verified, the LEDs blink several times and the output reads as follows:

```
-- Entering main() --


Starting Writing and Clearing Source and Destination Addresses.
Finished Writing and Clearing Source and Destination Addresses.
Starting DMA Transfer
Waiting.
DMA Transfer Complete, Verifying Destination Data
Destination Data is Correct
DMA Interrupt Cleared
Congratulations! DMA Operations Completed Successfully!


-- Exiting main() --
```

Running the ECC Software Application

The output on the HyperTerminal screen reads as follows:

```
-- Entering main() --


Starting ECC Testing:
 Starting Single Bit Error Testing:
  Writing to Reset Registers.....................................Done
  Writing to ECCCR...............................................Done
  Writing and Checking Memory Location............Done
  Checking ECCSR.................................................Done
  Checking ECCSEC................................................Done
 Starting Double Bit Error Testing:
  Writing to Reset Registers.....................................Done
  Writing to ECCCR...............................................Done
  Writing and Checking Memory Location............Done
```

```
  Checking ECCSR ..........................................Done
  Checking ECCDEC..........................................Done
Finished ECC Testing!


Writing to Reset Registers.....................................Done
-- Exiting main() --
```

References

DS326, *PLB Double Data Rate (DDR2) Synchronous DRAM (SDRAM) Controller (v1.01a)* Product Specification

DS472, *OPB Central DMA Controller (v1.00c)* Product Specification

Conclusion

This application note describes PLB DDR2 parameters, how to add ECC functionality to the PLB DDR2 memory controller, and how to utilize OPB Central DMA to test DMA operations inside the PLB DDR2 memory space. The reference system is built for the Xilinx ML410 Rev B board and includes two stand-alone software applications.

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
9/12/06	1.0	Initial Xilinx release.
6/7/07	1.1	Updated for EDK 9.1.02i