XAPP1121 (v1.0) October 9, 2008

Reference System: Optimizing Performance in PowerPC 440 Processor Systems

Author: James Lucero

Abstract

This reference system demonstrates improving system performance in the PowerPC® 440 Processor Block on the Virtex®-5 FXT FPGA. In this reference system optimizing performance for embedded DMA solutions is discussed. The XPS Central DMA master interface can be connected to the Processor Local Bus (PLB) v4.6 on either PLB Slave 0 (SPLB0) or PLB Slave 1 (SPLB1). Parameters for the XPS Central DMA are modified for the DMA engine to allow for parallel reads and writes through the crossbar. For HDMA, setting threshold values for interrupts and changing addresses in main memory for Buffer Descriptors (BD) and transmit/receive buffers is discussed. A simple loopback core is connected to the LocalLink interface on one HDMA. In addition, performance cores for PLB v.4.6 master interfaces and HDMA are included in the system to measure system performance before and after system optimizations.

Two stand-alone software applications are included to demonstrate DMA transactions for XPS Central DMA and HDMA to DDR2. In addition, during these DMA transactions, the performance is measured between certain conditions. For XPS Central DMA, the performance is measured between PLB_PaValid and when the XPS Central DMA provides an interrupt (DMA transactions are complete). For HDMA, the Tx and Rx channels are measured between the first frame's start of frame (SOF) and the last frame's end of frame (EOF).

This application note describes how to obtain latency numbers across the crossbar, obtain performance numbers before optimizing the system/software applications, optimizing the system/software application for performance and obtaining performance numbers with the optimized system. The Xilinx ML507 Rev A board is used for this reference system.

Included Systems

Included with this application note is two reference systems, V5_PPC440_nonopt_performance and V5_PPC440_opt_performance, built for the Xilinx ML507 Rev A board. The reference systems are available in the following ZIP files and are available at:

For the V5_PPC440_nonopt_perf_performance reference system, go to https://secure.xilinx.com/webreg/clickthrough.do?cid=113290.

For the V5_PPC440_opt_performance reference system, go to https://secure.xilinx.com/webreg/clickthrough.do?cid=113289.

Introduction

The PowerPC 440 processor block has many performance advantages compared to previous embedded solutions. For example, the crossbar switches between the Memory Interface Block (MIB) and the PLB Master (MPLB) where PLB v4.6 slaves are connected. Bus contention between PLB v4.6 slave peripherals and the memory controller connected to the Memory Interface Block (MIB) is eliminated by having separate ports on the processor block. Also bus contention between PLB v4.6 masters in the system is alleviated by providing two interfaces into the crossbar which access the MIB or MPLB.

Measuring performance in PPC® 440 systems is important to ensure the system is optimally built. Measuring performance and optimizing performance is discussed in this application note.

Hardware Requirements

The hardware requirements for this reference system are:

- Xilinx ML507 Rev A board
- Xilinx Platform USB or Parallel IV programming cable
- RS232 serial cable and serial communication utility (HyperTerminal)
- Xilinx Platform Studio 10.1.02
- Xilinx Integrated Software Environment (ISE) 10.1.02

Reference System Specifics

This system uses the PowerPC 440 processor block with a processor frequency of 400 MHz and the MIB frequency of 266 MHz. The processor block crossbar is set to 266 MHz. In addition, the MPLB, SPLB0, and SPLB1 ports of the crossbar are set to 133 MHz.

The reference system includes PPC440MC DDR2, LLDMA EXERCISER, XPS Central DMA, XPS BRAM, XPS UART16550, XPS GPIO, PERF DMA, PERF PLBV46 and XPS INTC.

Performance cores for PLB v4.6 masters and HDMA are connected to the system. These cores are described later in this application note. Also, the LLDMA EXERCISER which is a LocalLink loopback core is connected to the first HDMA port through the LocalLink interface on the processor block.

PPC440MC DDR2 is connected to the MIB of the processor block with a frequency of 266 MHz.

The slave connections of the two XPS Central DMA instances are connected to the PLB v4.6 instance connected to the MPLB. In addition, the XPS BRAM, PERF DMA, PERF PLBV46 instances, XPS GPIO, and XPS UART16550 cores are connected as slaves to the PLB v4.6 instance connected to the MPLB.

The master connections for the two XPS Central DMA instances are connected to the SPLB0 and SPLB1 ports on the processor block.

In addition, ChipScope PLB v4.6 IBA cores are added to SPLB0 and MPLB. Also, ChipScope ILA cores are added to the Memory Controller Interface (MCI) and the LocalLink interface. Adding these cores are not discussed in this application note.

See Figure 1 for the block diagram and Table 1 for the address map of the system.

Block Diagram

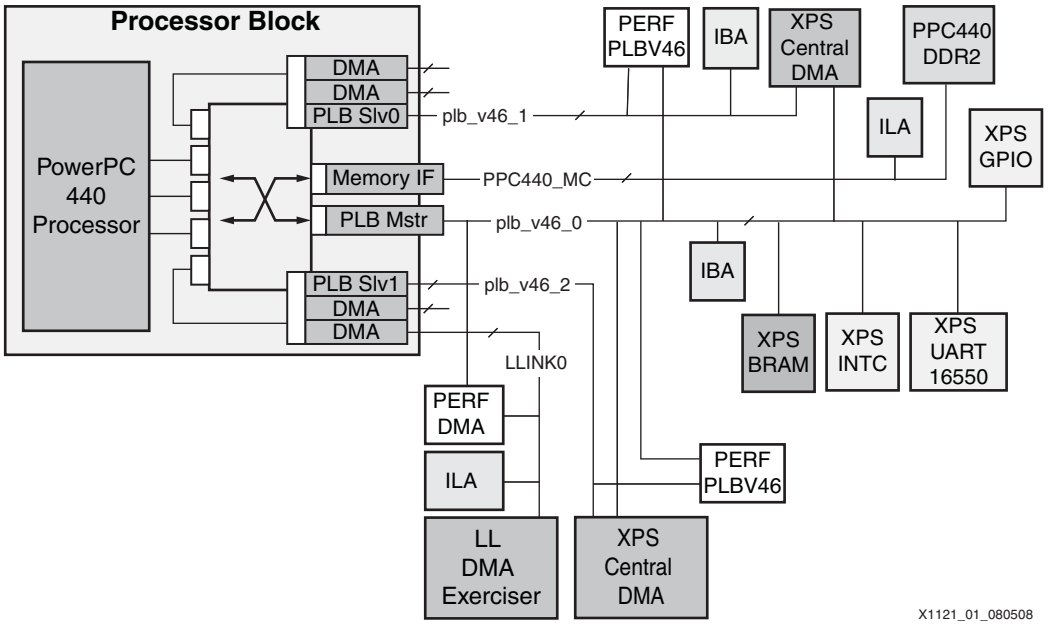

Figure 1: **Reference System Block Diagram**

Address Map

Table 1: **Reference System Address Map**

Peripheral	Instance	Base Address	High Address
ppc440mc_ddr2	ppc440_mc_ddr2_0	0x00000000	0x0FFFFFFF
xps_uart16550	xps_uart16550_0	0x40400000	0x4040FFFF
xps_gpio	LEDs_8Bit	0x40000000	0x4000FFFF
xps_bram_if_cntlr	xps_bram_if_cntlr_1	0xFFFF0000	0xFFFFFFFF
xps_central_dma	xps_central_dma_0	0x51000000	0x5100FFFF
xps_central_dma	xps_central_dma_1	0x52000000	0x5200FFFF
xps_intc	xps_intc_0	0x41200000	0x4120FFFF
perf_dma	perf_dma_0	0xB0000000	0xB000FFFF
perf_plbv46	perf_plbv46_0	0xC0000000	0xC000FFFF
perf_plbv46	perf_plbv46_1	0xC0010000	0xC001FFFF

Software Applications

LL PERF

This software application demonstrates a LocalLink transfer in which the transmit data is looped back to the receive channel by means of a loopback core connected to the LocalLink interface.

The user defines the size of the packet, the amount of BDs, and the coalescing threshold value for interrupts. In this software application, the packet size is 2K, the amount of buffer descriptors

is 128. The threshold value is set to 1 for the unoptimized system's software application which is changed in the optimized system's software application.

An instance of the `struct dma_chan_parms` is instantiated for the first HDMA block used in the system. The base address of the HDMA block is set, the PERF DMA core base address is set, the Tx and Rx interrupt vectors are set, the Tx and Rx BD base and high addresses are set, and the Tx and Rx buffer base addresses are set.

The Tx and Rx buffer descriptors' base and high address values depend on how many BDs are requested from the software application. With this software application, the value is set to 128 BDs for both channels. Each buffer descriptor takes 0x40 of memory space which contains the next pointer information, current buffer address, current buffer length, status/control and application specific data. In this example, the TX BD base and high addresses are 0x01000000 and 0x01001FFF. The total address range is 0x1FFF. Dividing the total address range by 0x40 leaves 128 BDs for the Tx side. The same procedure is for the Rx channel which also contains 128 BDs.

In the main function, the DMA instance is started based upon the base address for the DMA engine (DCR address for HDMA).

In the TxSetup function calculates the number of BDs depending on the base and high address set for the Tx BDs and creates a ring. The fields for the first BD are cleared then every BD in the ring is copied with the same data for the fields as the first BD. Then the interrupt for the Tx channel is started and the TX ring is started.

The RxSetup function calculates the number of buffer descriptors depending on the base and high address set for the Rx BDs and creates a ring. The fields for the first BD are cleared and then every BD in the ring is copied with the same data for the fields as the first BD. The Rx available BDs in the Rx ring are reserved. Every BD in the ring is assigned with the buffer address, SOP and EOP flags, length, and the buffer is assigned to every BD. In this case, every buffer descriptor shares the same buffer. The buffer descriptors are written to memory. The interrupts for the Rx channel is started and the Rx ring is started. After this process, the Rx channel is ready to receive packets.

In the SetupIntrSystem, the interrupt system is set up for the Rx and Tx channels. Once an interrupt occurs on the Rx channel, the interrupt routine ensures that no error occurred, and releases the BDs back into memory. Once an interrupt occurs on the Tx channel, the interrupt routine ensures that no error occurred during transmit and then releases the BDs back into memory.

The PERF DMA core is initialized by writing to the appropriate bit in the control register. The core is set up to start counting on the first SOF and stop counting on the last EOF which depends on the amount of buffer descriptors transferred.

In the SendPacket function, the packet is created with a simple pattern. The 128 Tx BDs in the Tx ring are reserved in memory. Every BD is assigned with the same buffer address, SOP and EOP flags, length, and the packet. The BDs are written to memory which starts the DMA transfer.

After the DMA transaction is started, the software application polls the status register of the PERF DMA core to monitor when the transfer is done.

When the DMA transaction is complete, the software reports the results of the DMA transfer in MBps.

PLBV46 DMA PERF

This software application demonstrates a simple DMA operation from a source address to a destination address.

The user defines the size of the DMA transfer, and the setup of the DMA operation. With this software application, the size of the DMA transfer is 8 MB and the two XPS Central DMA instance transactions are executed at the same time.

Two instances of the `struct dma_chan_parms` are set up for the two XPS Central DMA instances. The Device Id for the XPS Central DMA is set, the base address of the PERF PLBV46 instance is set, the Master Id for the XPS Central DMA master interface is set, the interrupt vector for the XPS Central DMA is set, and the source and destination addresses for the DMA transaction are set.

At the start of the software application, the memory region assigned to the source address is cleared, then data is written to the memory region of the source address. The XPS Central DMA cores are initialized and then set up to use interrupts.

The PERF PLBV46 instances are initialized by writing to the appropriate bit in the control register which includes the Master Id for the master interface. The core is set up to start counting on the assertion of PLB_PaValid signal and the proper master ID value and stops counting when an interrupt occurs from the XPS Central DMA core.

DMA operations start when the source base address and destination base address are written to the appropriate XPS Central DMA instance register. When the transfer is complete, an interrupt occurs.

When the DMA transactions are complete, the software reports the results of both XPS Central DMA transfers in Mbytes/s.

Reference System Specifics

The PERF PLBV46 and PERF DMA cores are discussed. Latency through the crossbar between SPLB0 and MPLB/MIB is analyzed. In addition, with the first unoptimized system, performance numbers from the software application are examined. To optimize the system/software applications, XPS Central DMA parameters are set and the ll perf software application is modified. With the optimized system/modified ll perf software application, performance numbers are examined.

Description of PERF PLBV46 and PERF DMA Cores

PERF PLBV46 Core

The PERF PLBV46 core measures simultaneous reads/writes from a single master on the PLB v4.6. This is beneficial when measuring a DMA engine connected to the PLB v4.6 like the XPS Central DMA. The core consists of a slave interfaces for the core's registers and a PLB v4.6 monitor interface to monitor PLB_PaValid where the master interface is connected. The core contains one parameter C_MSTID_WIDTH which needs to be set. This parameter is set to log2 of the total number of masters on the bus.

In this system, the two PERF PLBV46 core's bus monitor interfaces are connected to SPLB0 and SPLB1. In both cases, the slave interfaces are both connected to MPLB. The C_MSTID_WIDTH is set to 1.

The appropriate PLB_PaValid signal is monitored based upon the MasterId which is set in the control register. The core measures between the assertion of PLB_PaValid to the assertion of the interrupt from the core which indicates that the master transactions is compete. The time between these two events are stored in the counter register.

The PERF PLBV46 driver has several driver functions to control the hardware.

The Setup_PERF_PLBV46 function's arguments are the base address for the PERF PLBV46 core and the Master_Id. The function sets the control register. This function should be called before executing the DMA transaction.

The Poll_Done_PERF_PLBV46 function's argument is the base address for the PERF PLBV46 core. The function polls the status register until the DMA transaction is complete. This function should be called immediately after executing the DMA transaction function.

The PERF_PLBV46_Transfer function's arguments are the base address for the PERF PLBV46 core, the period of the core in NS, and the total length of the transfer. This function reports the results of the transaction in Mbps. This function should be called after executing the Poll_Done_PERF_PLBV46 function.

An example of using this core is provided in the plbv46_dma_perf software application which uses two instances of the core for the two XPS Central DMA instances.

The following table shows the registers for this core.

Table 2: **PERF PLBV46 Registers**

Register Name	(offset from C_BASEADDR)	Access	Size in Bits	Description
Control Register	0x0	Read/Write	32	Bit0 - Look for transfer. Bit1-n - Master_Id
Status Register	0x4	Read	32	Bit0 - Busy Bit1 - Done
Counter	0x8	Read	32	Counter value.
Software Reset	0x100	Write	32	Reset core by writing 0xA.

PERF DMA Core

The PERF DMA core measures throughput on both the Tx and Rx channels on the LocalLink interface connected to a single HDMA block. The core consists of a slave PLB v4.6 interface for the PERF DMA registers. In addition, several signals must be connected in the system to monitor the LocalLink signals. These include the LL_CLK, TX SOF, TX EOF, RX SOF, and RX EOF.

In this system the PERF DMA slave interface is connected to MPLB. In addition, the TX SOF, TX SOF, RX SOF and RX EOF are connected based upon the default net names with the LocalLink bus interface connection between the PPC440 wrapper and the LLDMA EXERCISER core. The LL_CLK is connected to the LocalLink clock in the system.

For Tx, the core measures between the first TX SOF and the last TX EOF. The last frame is determined by the Number of Frames register in the core which applies to both the Tx and Rx measurements. The time between these two events are stored into the Tx Counter register.

For Rx, the core measures between the first RX SOF and the last RX EOF. As with Tx, the last frame is determined by the Number of Frames register in the core which applies to both the Tx and Rx measurements. The time between these two events are stored into the Rx Counter register.

The PERF DMA driver has several driver functions to control the hardware.

The Setup_PERF_DMA function's arguments are the base address for the PERF DMA and the number of frames to measure. The control register is set to look for the transaction and the Number of Frames register is set. This function should be called before starting the DMA transaction (before putting buffer descriptors into hardware).

The Poll_Done_PERF_DMA function's argument is the base address for the PERF DMA core. The function polls the status register until the DMA transaction is complete. This function should be called immediately after executing the DMA transaction function.

The Tx_Transfer function's arguments are the base address for the PERF DMA core, the period of the core in nanoseconds, and the total length of the transfer. This function reports the results of the Tx transaction in Mbps. This function should be called after executing the Poll_Done_PERF_DMA function.

The Rx_Transfer function's arguments are the base address for the PERF DMA core, the period of the core in NS, and the total length of the transfer. This function reports the results of the Tx transaction in Mbps. This function should be called after executing the Poll_Done_PERF_DMA function.

The following table shows the registers for this core.

Table 3: **PERF DMA Registers**

Register Name	(offset from C_BASEADDR)	Access	Size in Bits	Description
Control Register	0x0	Read/Write	32	Bit0 - Look for transfer.
Number of Frames	0x4	Read/Write	32	Amount of frames to measure.
Status Register	0x8	Read	32	Bit0 - Busy Rx Bit1 - Done Rx Bit2 - Busy Tx Bit3 - Done Tx
Tx Counter	0xC	Read	32	Tx counter value.
Rx Counter	0x10	Read	32	Rx counter value.
Software Reset	0x100	Write	32	Reset core by writing 0xA.

Executing the Unoptimized Reference System in Hardware

Using HyperTerminal or a similar serial communications utility, map the operation of the utility to the physical COM port to be used. Then connect the UART of the board 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**. This is necessary to see the results from the software application.

The HyperTerminal settings are shown in Figure 2,

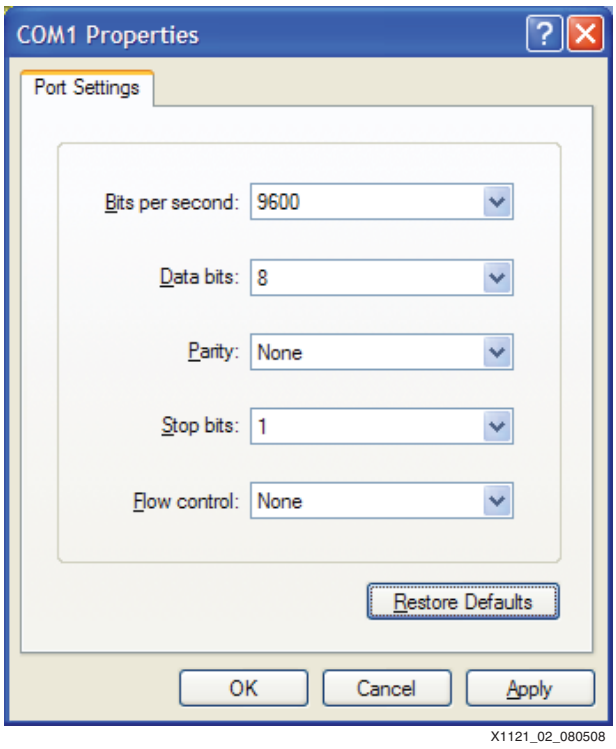

Figure 2: **HyperTerminal Settings**

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

To execute the system using files in the `ready_for_download` directory in the V5_PPC440_nonopt_performance project root directory, follow these steps:

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

   ```
   impact -batch xapp1121.cmd
   ```

3. Invoke XMD and connect to the PowerPC 440 processor by using the following command:

   ```
   xmd -opt xapp1121.opt
   ```

4. Download the executables by using the following command depending on the software application:
   - **`dow plbv46_dma_perf.elf`**
   - **`dow ll_perf.elf`**

Enter in the `run` command to run the software application.

Executing the Reference System from XPS for Hardware

To execute the system (V5_PPC440_nonopt_performance) for hardware using XPS, follow these steps:

1. Open `system.xmp` in XPS.
2. Use **Hardware→Generate Bitstream** to generate a bitstream for the system.
3. To compile the software applications, select **Software→Build All User Applications.**
4. Download the bitstream by selecting **Device Configuration→Download Bitstream**.

5. Invoke XMD by selecting **Debug→Launch XMD...** .
6. Download the executables by using the following command depending on the software application:
   - **dow plbv46_dma_perf/plbv46_dma_perf.elf**
   - **dow ll_perf/ll_perf.elf**
7. Enter in the `run` command to run the software application.

Running the Software Applications

The results from the unoptimized software application will be discussed after optimizing the system.

Running the PLBV46 DMA PERF Software Application

The following output is seen after running the software application.

Note: Numbers are slightly different after each run based upon many factors.

```
Starting XPS Central DMA Transfer(s):
XPS Central DMA 0 131.970695
XPS Central DMA 1 131.969636
Test Completed!
```

Running the LL PERF Software Application

The following output is seen after running the software application.

Note: Numbers are slightly different after each run based upon many factors.

```
Starting HDMA Transfer(s) to DDR2:

HDMA0 Tx 394.127465
HDMA0 Rx 394.088087

Test Completed!
```

Analyzing Latency Through the Crossbar

In this section, command latency will be analyzed between the first master transaction between XPS Central DMA on SPLB0 and MPLB/MIB. In ChipScope, the latency is measured between one tick after SPLB0_PaValid and MPLB_PaValid/McAddressValid.

In order to measure between SPLB0 and the MIB, the software application is changed to support only 1 DMA transfer and the program start address for the software application is changed to run from XPS BRAM. This ensures no other transactions like processor fetches are occurring to the MIB.

To measure from SPLB0 to the MPLB, the software application is changed to execute DMA transactions to XPS BRAM. Then the program start address is changed back to DDR2.

Measuring between SPLB0 to MCI

1. In the PLBV46 DMA PERF software application, change **#define NUMBER_OF_DMAS 2** to **#define NUMBER_OF_DMAS 1**. This change can be done in XPS or another text editor.
2. In the XPS/Applications tab, right click on Project: plbv46_dma_perf and select **Set Compiler Options...** . For **Program Start Address**, set this value to **0xFFFF0000**.
3. Right click on Project: plbv46_dma_perf and select **Build Project**.
4. Open ChipScope Analyzer.
5. Click on the Open Cable / Search JTAG Chain button and then click **OK** on the ChipScope Pro Analyzer dialog box.

6. Select **File→Open Project...**, select **No** to save the new project, and select the `ppc440_performance.cpj` in the cpj directory of the V5_PPC440_nonopt_performance project.

   In the ChipScope project, Table 4 shows the associated ChipScope cores.

Table 4: **ChipScope Icon Cores**

Unit	BUS
UNIT:0 MyILA0	MPLB
UNIT:1 MyILA1	SPLB
UNIT2: MyILA2	HDMA
UNIT3: MyILA3	MCI

7. For UNIT:3 MyILA3 (MCI), set the trigger by **Trigger Setup→Run**. The trigger is setup for the input PLB_PaValid signal from plb_v46_1.
8. The software application has been compiled. In XMD download plbv46_dma_perf by the following command and run the executable:
   - `dow plbv46_dma_perf/plbv46_dma_perf.elf`
9. In the ChipScope project, select the Waveform for UNIT:3 MyILA3 (MCI). The system will display the waveform shown in Figure 3..

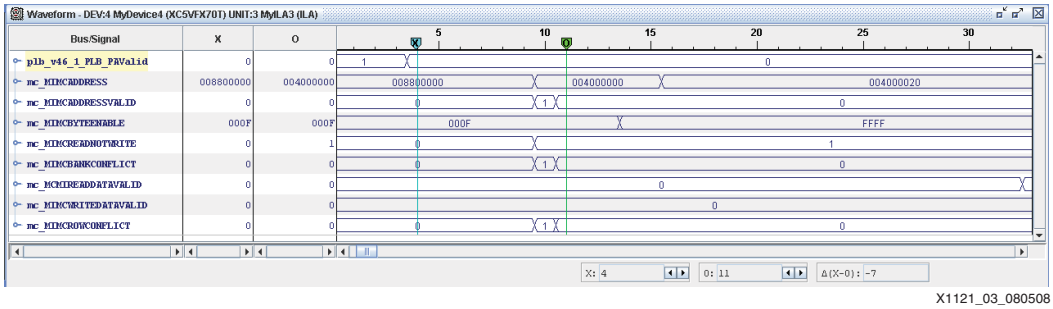

Figure 3: **ChipScope MIB Latency**

To measure the command latency, place the X cursor one tick after the deassertion of plb_v46_1_PLB_PAValid and place the O cursor one tick after the deassertion of mc_MIMCADDRESSVALID signal. The latency between the SPLB0/SPLB1 to the MCI is 7 MIB clocks which is 3.5 PLB clocks. This is with the 266 MHz MIB/Crossbar and the 133 MHz PLB clock ratio.

Measuring Between SPLB0 to MPLB

1. In the plbv46_dma_perf software application, change **#define DMA0_BRAM 0** to **#define DMA0_BRAM 1**. This change can be done in XPS or a text editor.
1. In XPS/Applications tab, right click on Project: plbv46_dma_perf and select **Set Compiler Options...** . For **Program Start Address**, set the value back to **0x00000000**.
2. Right click on Project: plbv46_dma_perf and select **Build Project**.
3. For UNIT:0 MyILA0 (MPLB), set the trigger by **Trigger Setup→Run**. The trigger is setup for the input PLB_PaValid signal from plb_v46_1.
4. Since the software application has been compiled, in XMD, download plbv46_dma_perf by the following command and run the executable:
   - `dow plbv46_dma_perf/plbv46_dma_perf.elf`

5. In the ChipScope project, select the Waveform for UNIT:0 MyILA0 (MPLB). The system will display the waveform shown in Figure 4.

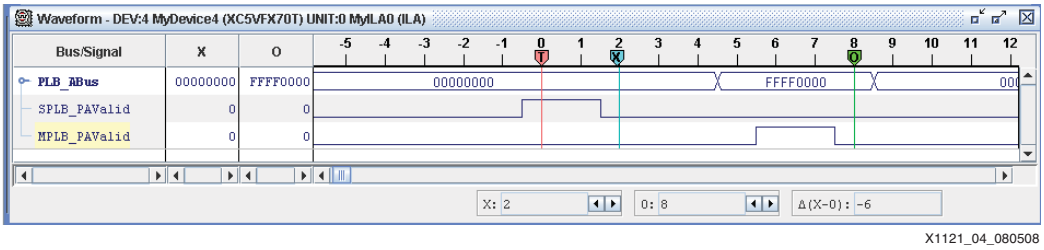

Figure 4: **ChipScope MPLB Latency**

To measure latency, place the X cursor one tick after the deassertion of SPLB_PAValid and place the O cursor one tick after the deassertion of the MPLB_PAValid signal. The latency between the SPLB0/SPLB1 to the MPLB is 6 PLB clocks.

Optimizing XPS Central DMA and LL PERF Software Application

This section covers setting XPS Central DMA parameters to allow for overlapping reads and writes to increase performance with the XPS Central DMA core. With the LL PERF software application, locations of Tx and Rx buffer descriptors and buffers are modified to use different banks and the interrupt coalescing value is changed to increase performance.

Setting XPS Central DMA Parameters

1. In the System Assembly View, right click on **xps_central_dma_0** and select **Configure IP...** . In the User/All tab, select **FIFO Depth** to **48**, **Read Burst Size** to **16** and **Write Burst Size** to **8**. These parameter settings allow for overlapped reads and writes on the PLB.

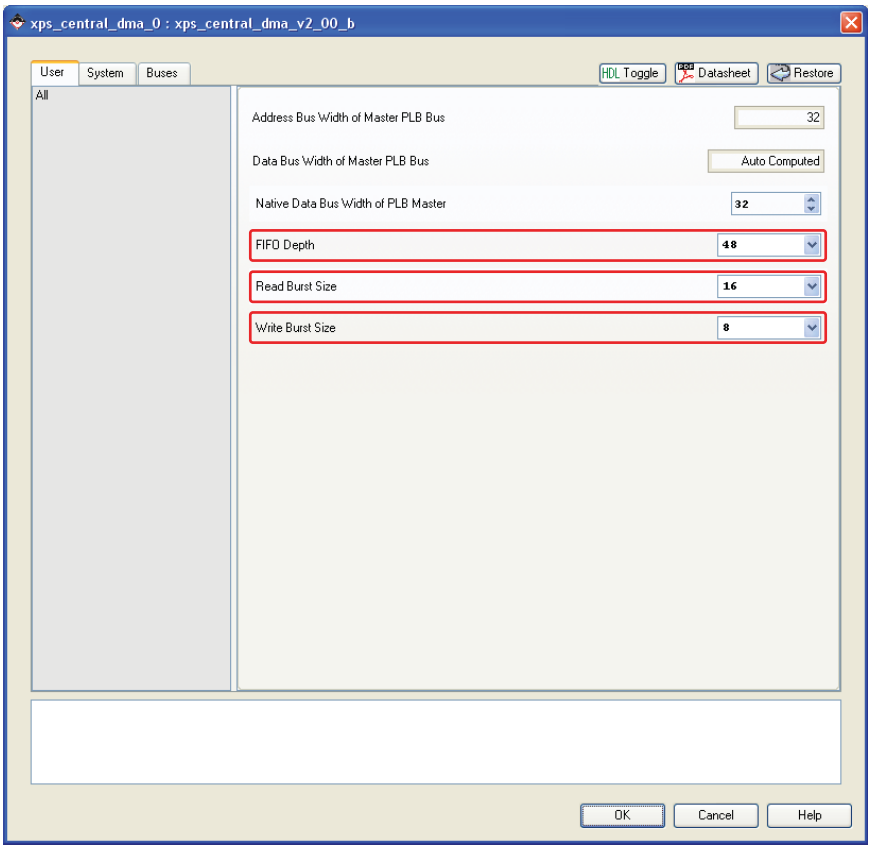

Figure 5: **XPS Central DMA Parameters**

2. In the System Assembly View, right click on **xps_central_dma_1** and select **Configure IP...** . In the User/All tab, select **FIFO Depth** to **48**, **Read Burst Size** to **16** and **Write Burst Size** to **8**. These parameter settings allow for overlapped reads and writes on the PLB.

Modifying LL PERF Software Application

Note: Ensure that the original LL PERF software application from the unoptimized system is used for this section.

1. The original software application creates an interrupt that needs servicing after every frame. The interrupt servicing can occur after a set of frames by changing the coalescing threshold define value (value between 1-255 frames). Change the `#define COALESCING_THRESHOLD 1` to `#define COALESCING_THRESHOLD 32`.
2. With the ML507 board, 4 memory banks are available. The Tx and Rx buffer descriptors and buffers will be split between these 4 banks instead of 1 bank as the original software application. Since 4 banks can be open at one time, the efficiency of the traffic pattern increases since opening rows in the same bank causes conflicts. With the PPC440MC DDR2, four banks can be open at the same time which decreases the amount of row conflicts with this setup.

   Scroll down to the main function on line 928. When defining the struct change the following segments of code.

   Change line 960 from:

   ```
   dma_struct[0].RX_BD_BASE = XPAR_PPC440_MC_DDR2_0_MEM_BASEADDR +
   0x01002000;
   ```

   to:

   ```
   dma_struct[0].RX_BD_BASE = XPAR_PPC440_MC_DDR2_0_MEM_BASEADDR +
   0x04002000;
   ```

   Change line 962 from:

   ```
   dma_struct[0].RX_BD_HIGH = XPAR_PPC440_MC_DDR2_0_MEM_BASEADDR +
   0x01003FFF;
   ```

   to:

   ```
   dma_struct[0].RX_BD_HIGH = XPAR_PPC440_MC_DDR2_0_MEM_BASEADDR +
   0x04003FFF;
   ```

   Change line 965 from:

   ```
   dma_struct[0].TX_BUFFER_BASE = XPAR_PPC440_MC_DDR2_0_MEM_BASEADDR +
   0x02600000;
   ```

   to:

   ```
   dma_struct[0].TX_BUFFER_BASE = XPAR_PPC440_MC_DDR2_0_MEM_BASEADDR +
   0x08600000;
   ```

   Change line 967 from:

   ```
   dma_struct[0].RX_BUFFER_BASE = XPAR_PPC440_MC_DDR2_0_MEM_BASEADDR +
   0x03610000;
   ```

   to:

   ```
   dma_struct[0].RX_BUFFER_BASE = XPAR_PPC440_MC_DDR2_0_MEM_BASEADDR +
   0x0C610000
   ```

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

To execute the system using files in the `ready_for_download` directory in the V5_PPC440_opt_performance project root directory, follow these steps:

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

   ```
   impact -batch xapp1121.cmd
   ```

3. Invoke XMD and connect to the PowerPC 440 processor by using the following command:

   ```
   xmd -opt xapp1121.opt
   ```

4. Download the executables by using the following command depending on the software application:
   - **`dow plbv46_dma_perf.elf`**
   - **`dow ll_perf.elf`**

Enter in the `run` command to run the software application.

Executing the Reference System from XPS for Hardware

To execute the system (V5_PPC440_opt_performance) for hardware using XPS, follow these steps:

1. Open `system.xmp` in XPS.
2. Use **Hardware→Generate Bitstream** to generate a bitstream for the system.
3. To compile the software applications, select **Software→Build All User Applications**.
4. Download the bitstream by selecting **Device Configuration→Download Bitstream**.
5. Invoke XMD by selecting **Debug→Launch XMD...** .
6. Download the executables by using the following command depending on the software application:
   - **`dow plbv46_dma_perf/plbv46_dma_perf.elf`**
   - **`dow ll_perf/ll_perf.elf`**
7. Enter in the `run` command to run the software application.

Running the Software Applications

Running the PLBV46 DMA PERF Software Application

The following output should be seen from running the software application.

Note: Numbers are slightly different after each run based many factors.

```
Starting XPS Central DMA Transfer(s):
XPS Central DMA 0 178.650407
XPS Central DMA 1 178.649322
Test Completed!
```

Compared to the unoptimized system, performance increased ~25%.

Running the LL PERF Software Application

Note: Numbers are slightly different after each run based upon many factors.

The following output should be seen from running the software application.

```
Starting HDMA Transfer(s) to DDR2:

HDMA0 Tx 476.051263
HDMA0 Rx 476.064032

Test Completed!
```

Compared to the unoptimized LL PERF software application, performance increased by ~20%.

References

1. UG200 *Embedded Processor Block in Virtex-5 FPGAs Reference Guide*

Notice of Disclaimer

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
10/9/08	1.0	Initial Xilinx release.