XAPP1003 (v1.1) September 2, 2008

Reference System: PowerPC 440 System Simulation

Author: James Lucero

Abstract

This reference system demonstrates the functionality of the PowerPC® 440 Processor Block on the Virtex® -5 FXT FPGA. Processor Local Bus (PLB) v4.6 peripherals are connected to PLB v4.6 instances on the PLB Master (MPLB), and PLB Slave 0 (SPLB0). The Device Control Register (DCR) port is connected to DCR peripherals such as an interrupt controller, and connections are made to the Hard DMA (HDMA) port. This system includes common peripherals such as Ethernet, DDR2, DMA, Interrupt Controller and RS232.

Several stand-alone software applications are provided to verify the functionality of the peripherals and the PowerPC 440 Processor Block. Applications include hello_uart, hello_gpio, hello_dma, hello_temac, and TestApp_Memory.

This application note describes how to set up the simulation environment for the system, execute the simulation, and add PLB v4.6 peripherals to the existing system. In addition, the Xilinx ML507 Rev A board is used to verify the system in hardware on the Virtex-5 FXT FX70TFF1136-1 FPGA.

Included Systems

Included with this application note is one reference system, V5_PPC440_system_simulation, built for the Xilinx ML507 Rev A board. The reference system is available in the ZIP file and is available at:

https://secure.xilinx.com/webreg/clickthrough.do?cid=107508.

Introduction

The PowerPC 440 processor block has many embedded solution advantages. For example, it has higher bandwidth to external memory based on the dedicated Memory Controller (MC) interface. The PLB interfaces on the processor block eliminates bus arbiters in many systems.

Simulation Requirements

Simulation environment is verified with:

- Red Hat Enterprise Linux 3.0 or Windows XP
- ModelTech SE 6.3c with VHDL and Verilog co-simulation capability
- Xilinx Platform Studio 10.1.02
- Xilinx Integrated Software Environment (ISE®) 10.1.02

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 Memory Interface Block (MIB) frequency of 266 MHz. The processor block crossbar is set to 266 MHz. In addition, the MPLB, SPLB0, SPLB1and DCR ports are set to 133 MHz. For more information about valid clocking ratios and the PLL clocks controlling the processor block, refer to *Embedded Processor Block in Virtex-5 FPGAs Reference Guide*. For more information about port connections and parameter settings in the system, refer to the *PLB v3.4 and OPB to PLB v4.6 System and Core Migration User Guide* under the PPC405 - PPC440 System Migration chapter.

The reference system includes PPC440MC DDR2, XPS LL TEMAC, XPS Central DMA, XPS BRAM, XPS UART16550, XPS GPIO, and DCR INTC.

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

The slave connection of the XPS Central DMA is connected to the PLB v4.6 instance connected to the MPLB. In addition, the XPS BRAM, XPS LL TEMAC, and XPS UART16550 cores are connected as slaves to the PLB v4.6 instance connected to the MPLB. The LocalLink connection of the XPS LL TEMAC core is connected to the HDMA on the processor block.

The master connection on the XPS Central DMA is connected to the SPLB0 port on the processor block. The PowerPC 440 is setup for MPLB transactions to occur between the address ranges of `0x40000000` and `0xFFFFFFFF`. All slave peripheral addresses in this system are in this range.

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

Block Diagram

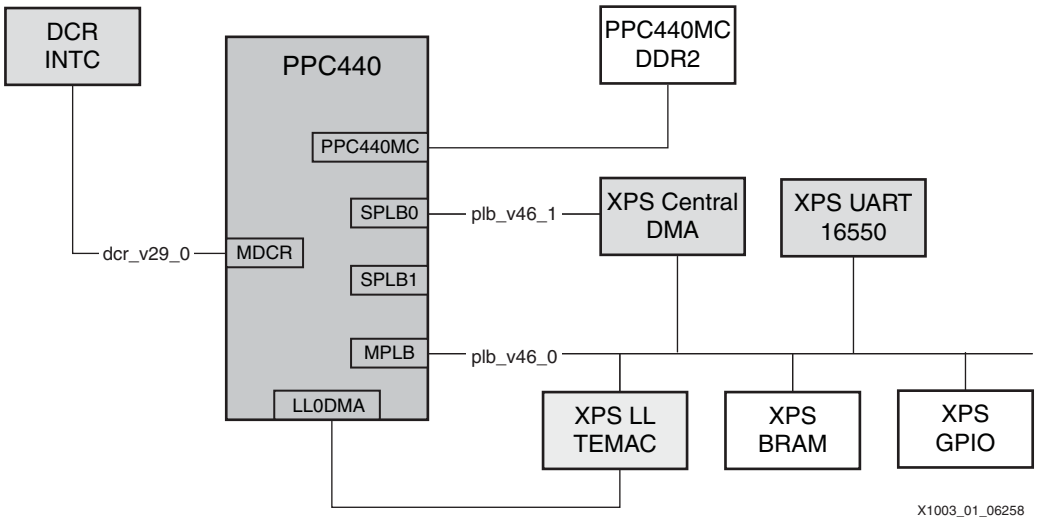


Figure 1: **Reference System Block Diagram**

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_gpio	ddr2_init	0x40040000	0x4004FFFF
xps_bram_if_cntlr	xps_bram_if_cntlr_1	0xFFFF0000	0xFFFFFFFF
xps_central_dma	xps_central_dma_0	0x51000000	0x5100FFFF
xps_ll_temac	xps_ll_temac_0	0x51200000	0x5120FFFF
dcr_intc	dcr_intc_0	0b1000000000	0b1011111111

Software Applications

Hello Uart Software Application

This software application tests functionality of the XPS UART16550. It prints `Hello World!` to the simulated RS232 terminal and to the RS232 terminal for hardware.

This software application is run out of DDR2 external memory using the PPC440MC DDR2 memory controller.

Hello Gpio Software Application

This software application tests the functionality of the XPS GPIO.

The external GPIO pins will represent the binary counts from 0 to 8 for simulation. The software application ends with the external GPIO pins going back to 0. No output is shown on the simulation RS232 terminal and the RS232 terminal for hardware.

In hardware, the eight LEDs will blink from left to right six times.

This software application is run out of DDR2 external memory using the PPC440MC DDR2 memory controller.

TestApp Memory Software Application

This software application tests the functionality of the XPS BRAM.

The TestApp Memory software application will run basic memory tests (32-bit,16-bit, 8-bit tests) on the XPS BRAM such as XUT_ALLMEMTESTS, XUT_INCREMENT, XUT_WALKONES, XUT_WALKZEROS, XUT_INVERSEADDR, and XUT_FIXEDPATTERN. The simulated RS232 screen and the RS232 terminal for hardware will output `PASSED!` or `FAILED!` for each of the three memory transaction sizes tested (32-bit,16-bit, and 8-bit).

This software application is run out of DDR2 external memory using the PPC440MC DDR2 memory controller.

Hello DMA Software Application

This software application tests the functionality of the XPS Central DMA, DCR INTC, and PPC440MC DDR2 memory controller.

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 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 XPS 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 that the data was transferred correctly. Also, the software application, by means of the interrupt handler, clears the interrupt caused by the DMA operation.

This software application is run out of DDR2 external memory using the PPC440MC DDR2 memory controller.

Hello Temac Software Application

This software tests the functionality of the XPS LL TEMAC and the HDMA on the processor block.

The software application initializes both the HDMA and XPS LL TEMAC instances. The code sets up buffer descriptors for both RX and TX in the PPC440MC DDR2 for the HDMA. The buffer sizes are smaller for simulation purposes. In addition, an interrupt service routine is started for the XPS LL TEMAC, DMA TX, and DMA RX interrupts. Because no PHY is provided in the simulation environment, the XPS LL TEMAC external signals are set up for loopback mode in the testbench. In hardware, the PHY is set up for loopback mode.

The software application demonstrates the functionality of the XPS LL TEMAC and HDMA by sending and receiving a single frame in SGDMA interrupt mode. The transmit frame is transferred from the HDMA to the XPS LL TEMAC through LocalLink. The receive frame is transferred from the XPS LL TEMAC to HDMA through LocalLink.

This software application is run out of DDR2 external memory using the PPC440MC DDR2 memory controller.

Setting the Simulation Environment

Compiling Libraries

1. Compile Unisim, Simprim, XilinxCoreLib, SecureIP, Smartmodel libraries and EDK libraries. Open XPS and select **Cancel** on the **Create new or open existing project dialog box**. Select **Simulation→Compile Simulation Libraries...**

Note: The hard blocks for this application note use SecureIP libraries and not Smartmodels libraries.

2. Follow the instructions to compile with the ModelSim simulator and select **Both VHDL and Verilog**. Both VHDL and Verilog simulation support are required for this reference system.

3. Set the options to compile the ISE libraries as shown in Figure 2.

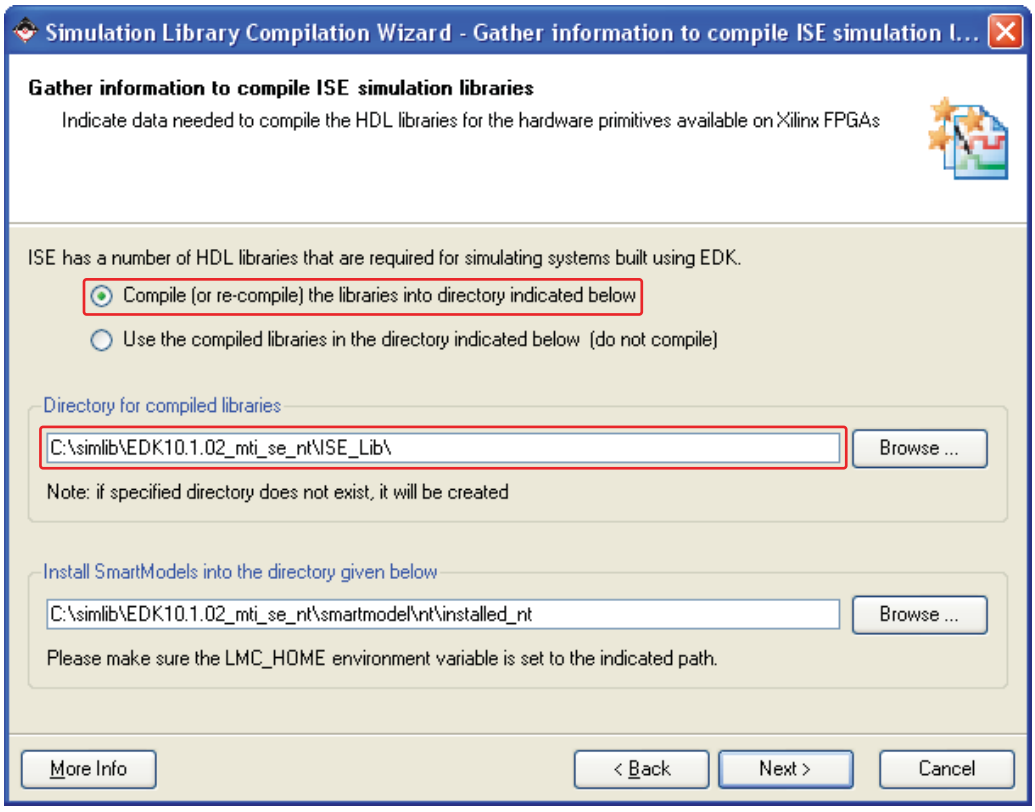


Figure 2: **Compiling ISE Libraries**

4. Set the options to compile the EDK libraries as shown in Figure 3.

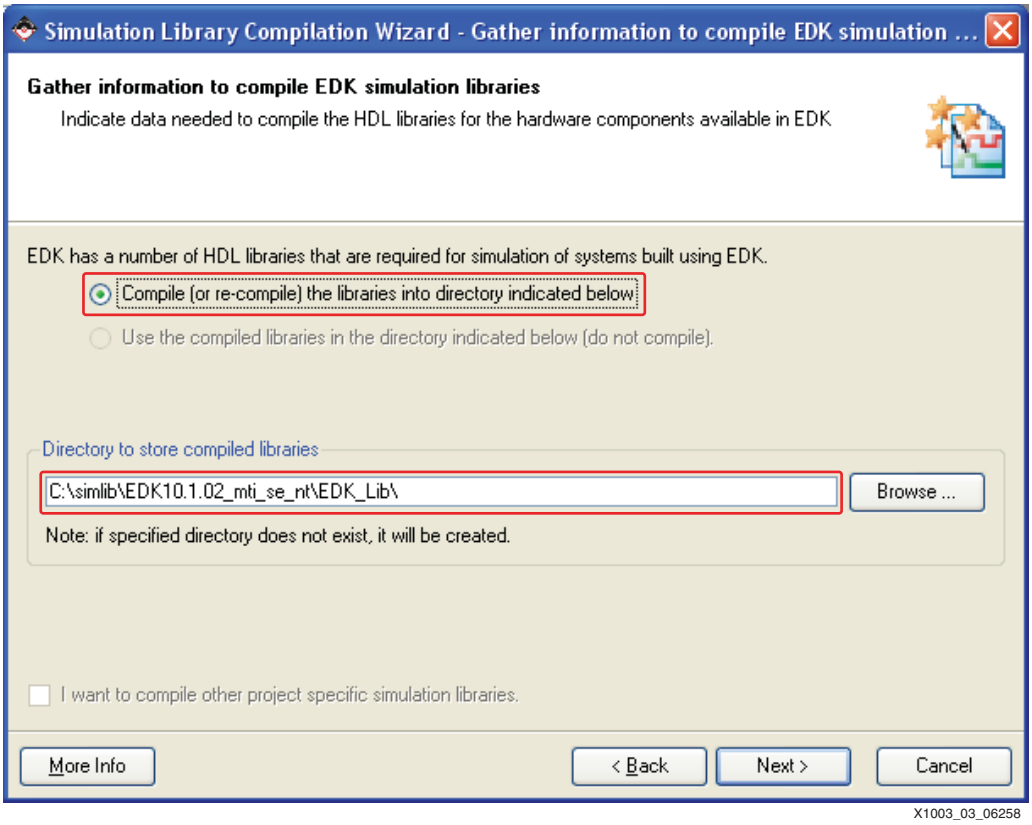


Figure 3: **Compiling EDK Libraries**

5. In the subsequent dialog box, select **Do not compile deprecated library elements**, then select **Compile >** to start the compilation process.

Note: This process may take a few hours depending on the machine being used.

6. Close XPS.

Setting Environmental Variables and ModelSim in Windows

1. Go to **Start→Control Panel→System→Advanced** and click on **Environment Variables**.
2. Set the following variables if necessary:
   a. MODELSIM <modelsim_install_directory>\modelsim.ini
3. Ensure that all of the EDK libraries and ISE libraries are mapped in `modelsim.ini`
   a. If the libraries are not mapped correctly, copy the library mappings from the `modelsim.ini` found in the compile EDK_Lib directory to the current `modelsim.ini` above the [Vcom] section.

An example `modelsim.ini` is provided in the `windows_setup` directory.

Setting Environmental Variables and ModelSim in Linux

1. Set the following variables:
   a. setenv MODELSIM <path to modelsim.ini script>/modelsim.ini
2. Ensure that the EDK libraries and ISE libraries are mapped in the `modelsim.ini`
   a. If the libraries are not mapped correctly, copy the library mappings from the compile EDK_Lib directory's modelsim.ini to the current modelsim.ini above the [Vcom] section.

An example `modelsim.ini` is provided in the `linux_setup` directory.

Simulation Directories and Files

Two simulation directories are used for Linux and Windows. The directories are `sim_lin/` and `sim_win/`.

Windows Simulation Files

The `sim_win/` directory contains the following pertinent files and directories:

- `256Mb_ddr2/` - Directory that contains Micron DDR2 memory model.
- `mem/` - Directory that contains the perl scripts to create the data files for the DDR2 memory model
- `system_tb_v.do` - ModelSim compile script for Verilog simulation.
- `system_tb.v` - Verilog testbench for system.
- `uart_rcvr.v` - For simulated RS232 terminal.
- `uart_rcvr_wrapper.v` - For simulated RS232 terminal.

Note: The simulated RS232 terminal won't show up in Windows. Output from the software application is monitored in the ModelSim terminal while the simulation is running.

Linux Simulation Files

The `sim_lin/` directory contains the following pertinent files and directories:

- `256Mb_ddr2/` - Directory that contains Micron DDR2 memory model.
- `mem/` - Directory that contains the perl scripts to create the data files for the DDR2 memory model
- `system_tb_v.do` - ModelSim compile script for Verilog simulation.
- `system_tb.v` - Verilog testbench for system.
- `uart_rcvr.v` - For simulated RS232 terminal.
- `uart_rcvr_wrapper.v` - For simulated RS232 terminal.

Executing the Simulation

Executing the Simulation from XPS

To generate and run the system simulation from EDK, follow these steps:

1. Open `system.xmp` in EDK.
2. Click on **Software→Launch Platform Studio SDK**
3. Select **Cancel** in the Application Wizard once SDK starts.
4. Select **Project→Build Automatically**. Before proceeding to the next step, ensure that all software applications are built. During this time the software projects and libraries are recompiled.

In the Applications/Software Projects tab in XPS, select **Make Project Active** and **Mark to Initialize BRAMs** for the software application that will be used in the simulation. These are the software projects with `_sim` in the software project name. This step is required for all applications regardless if they are run from either XPS BRAM or PPC440MC DDR2, so that the ELF will be converted to files that can be used for loading the external memory model.

> ***Caution!*** Only one project should be initialized while all other software applications projects are inactive.

Note: Step 5-8 are repeated to generate simulations for other software applications.

5. Use **Simulation→Generate Simulation HDL Files**.
6. Launch ModelSim with **Simulation→Launch HDL Simulator...**.
7. In ModelSim, execute the compile script for Verilog by the following command depending on the operating system:

```
do ../../sim_win/system_tb_v.do
```
or
```
do ../../sim_lin/system_tb_v.do
```

8. In ModelSim, execute the run command with the specified run time for the selected software application. For example:

   run 120us

Simulating Software Applications

Simulating the Hello Uart Software Application

This simulation should run for ~110 us. The resulting WLF file is provided at `WLFs/hello_uart.wlf`.

The fpga_0_RS232_Uart_1_sout_pin toggles between ~61 us and ~108 us, which provides output to the RS232.

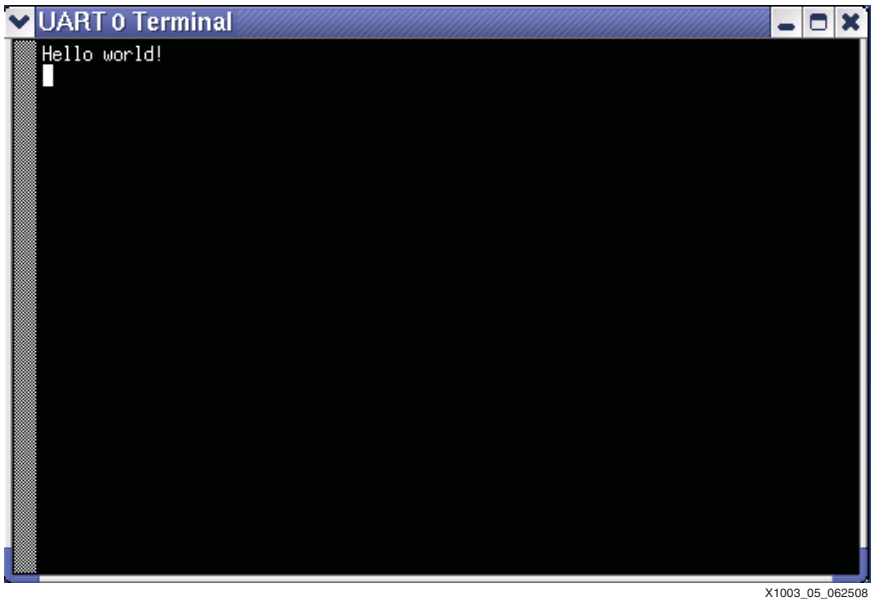
The XPS UART16550 starts transmitting output to the simulated RS232 terminal and the ModelSim transcript window, or both. In the Windows simulation environment, the XPS UART 16550 output is seen in the ModelSim transcript window as shown in Figure 4.

```
# UART     0 Transmitted Out Char = 0x48 = "H" @ Time = 51045000.0 ps
# UART     0 Transmitted Out Char = 0x65 = "e" @ Time = 55845000.0 ps
# UART     0 Transmitted Out Char = 0x6c = "l" @ Time = 60645000.0 ps
# UART     0 Transmitted Out Char = 0x6c = "l" @ Time = 65445000.0 ps
# UART     0 Transmitted Out Char = 0x6f = "o" @ Time = 70245000.0 ps
# UART     0 Transmitted Out Char = 0x20 = " " @ Time = 75045000.0 ps
# UART     0 Transmitted Out Char = 0x77 = "w" @ Time = 79845000.0 ps
# UART     0 Transmitted Out Char = 0x6f = "o" @ Time = 84645000.0 ps
# UART     0 Transmitted Out Char = 0x72 = "r" @ Time = 89445000.0 ps
# UART     0 Transmitted Out Char = 0x6c = "l" @ Time = 94245000.0 ps
```

X1003_04_062508

Figure 4: **Hello UART Output – Windows Simulation Environment**

Figure 5 shows the output in the Linux simulation environment.



X1003_05_062508

Figure 5: **Hello UART Output – Linux Simulation Environment**

Simulating the Hello Gpio Software Application

This simulation should run for ~66 us. The resulting WLF file is provided at `WLFs/hello_gpio.wlf`.

The LEDs start incrementing and return back to 0. The fpga_0_LEDs_8Bit_GPIO_IO_pin begins to increment at ~59 us. This signal increments from 0 to 8 during this time period and returns to 0 at ~60 us. No output is provided in either the Windows or the Linux simulation environments.

Simulating the TestApp Memory Software Application

This simulation should run for ~ 425 us. The resulting WLF file is provided at `WLFs/testapp_memory.wlf`.

The software application is currently set to do an increment test. The 32-bit memory test occurs between ~141 us and ~142 us. The 16-bit memory test occurs around ~260 us. The 8-bit memory test occurs around ~ 374 us. During these memory tests, monitor the XPS BRAM signals.

The software application outputs the status of the 32-bit, 16-bit, 8-bit memory tests. In the Windows simulation environment, the output is seen in the ModelSim transcript window. Some of this output is shown in Figure 6.

```
# UART     0 Transmitted Out Char = 0x0d = <special char> @ Time = 358185000.0 ps
# UART     0 Transmitted Out Char = 0x0a = <special char> @ Time = 362985000.0 ps
# UART     0 Transmitted Out Char = 0x20 = " " @ Time = 367785000.0 ps
# UART     0 Transmitted Out Char = 0x20 = " " @ Time = 372585000.0 ps
# UART     0 Transmitted Out Char = 0x52 = "R" @ Time = 377385000.0 ps
# UART     0 Transmitted Out Char = 0x75 = "u" @ Time = 382185000.0 ps
# UART     0 Transmitted Out Char = 0x6e = "n" @ Time = 386985000.0 ps
# UART     0 Transmitted Out Char = 0x6e = "n" @ Time = 391785000.0 ps
# UART     0 Transmitted Out Char = 0x69 = "i" @ Time = 396585000.0 ps
# UART     0 Transmitted Out Char = 0x6e = "n" @ Time = 401385000.0 ps
# UART     0 Transmitted Out Char = 0x67 = "g" @ Time = 406185000.0 ps
# UART     0 Transmitted Out Char = 0x20 = " " @ Time = 410985000.0 ps
# UART     0 Transmitted Out Char = 0x38 = "8" @ Time = 415785000.0 ps
# UART     0 Transmitted Out Char = 0x2d = "-" @ Time = 420585000.0 ps
# UART     0 Transmitted Out Char = 0x62 = "b" @ Time = 425385000.0 ps
# UART     0 Transmitted Out Char = 0x69 = "i" @ Time = 430185000.0 ps
# UART     0 Transmitted Out Char = 0x74 = "t" @ Time = 434985000.0 ps
# UART     0 Transmitted Out Char = 0x20 = " " @ Time = 439785000.0 ps
# UART     0 Transmitted Out Char = 0x74 = "t" @ Time = 444585000.0 ps
# UART     0 Transmitted Out Char = 0x65 = "e" @ Time = 449385000.0 ps
# UART     0 Transmitted Out Char = 0x73 = "s" @ Time = 454185000.0 ps
# UART     0 Transmitted Out Char = 0x74 = "t" @ Time = 458985000.0 ps
# UART     0 Transmitted Out Char = 0x2e = "." @ Time = 463785000.0 ps
# UART     0 Transmitted Out Char = 0x2e = "." @ Time = 468585000.0 ps
# UART     0 Transmitted Out Char = 0x2e = "." @ Time = 473385000.0 ps
```

X1003_06_062508

Figure 6: **TestApp Memory Output – Windows Simulation Environment**

Figure 7 shows the output in the Linux simulation environment.

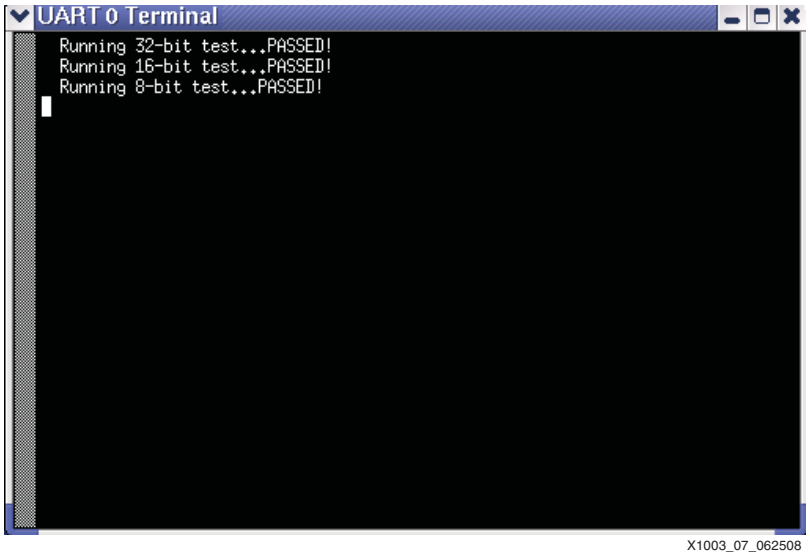


Figure 7: **TestApp Memory Output – Linux Simulation Environment**

Simulating the Hello DMA Software Application

This simulation should run for ~495 us. The resulting WLF file is provided at `WLFs/hello_dma.wlf`.

The XPS Central DMA slave registers are set. The writing and clearing of the source and destination addresses occurs. DMA transactions occur between ~187 us and ~188 us through the XPS Central DMA by watching the XPS Central DMA and external memory models signals. The DMA interrupt signal is cleared at ~197 us. In the Windows simulation environment, the output is seen in the ModelSim transcript window after a successful DMA transaction. Some of this output is shown in Figure 8.

```
# UART    0 Transmitted Out Char = 0x44 = "D" @ Time = 444165000.0 ps
# UART    0 Transmitted Out Char = 0x4d = "M" @ Time = 448965000.0 ps
# UART    0 Transmitted Out Char = 0x41 = "A" @ Time = 453765000.0 ps
# UART    0 Transmitted Out Char = 0x20 = " " @ Time = 458565000.0 ps
# UART    0 Transmitted Out Char = 0x49 = "I" @ Time = 463365000.0 ps
# UART    0 Transmitted Out Char = 0x6e = "n" @ Time = 468165000.0 ps
# UART    0 Transmitted Out Char = 0x74 = "t" @ Time = 472965000.0 ps
# UART    0 Transmitted Out Char = 0x65 = "e" @ Time = 477765000.0 ps
# UART    0 Transmitted Out Char = 0x72 = "r" @ Time = 482565000.0 ps
# UART    0 Transmitted Out Char = 0x72 = "r" @ Time = 487365000.0 ps
# UART    0 Transmitted Out Char = 0x75 = "u" @ Time = 492165000.0 ps
# UART    0 Transmitted Out Char = 0x70 = "p" @ Time = 496965000.0 ps
# UART    0 Transmitted Out Char = 0x74 = "t" @ Time = 501765000.0 ps
# UART    0 Transmitted Out Char = 0x20 = " " @ Time = 506565000.0 ps
# UART    0 Transmitted Out Char = 0x43 = "C" @ Time = 511365000.0 ps
# UART    0 Transmitted Out Char = 0x6c = "l" @ Time = 516165000.0 ps
# UART    0 Transmitted Out Char = 0x65 = "e" @ Time = 520965000.0 ps
# UART    0 Transmitted Out Char = 0x61 = "a" @ Time = 525765000.0 ps
```

X1003_08_062508

Figure 8: **Hello DMA Output – Windows Simulation Environment**

Figure 9 shows the output in the Linux simulation environment.

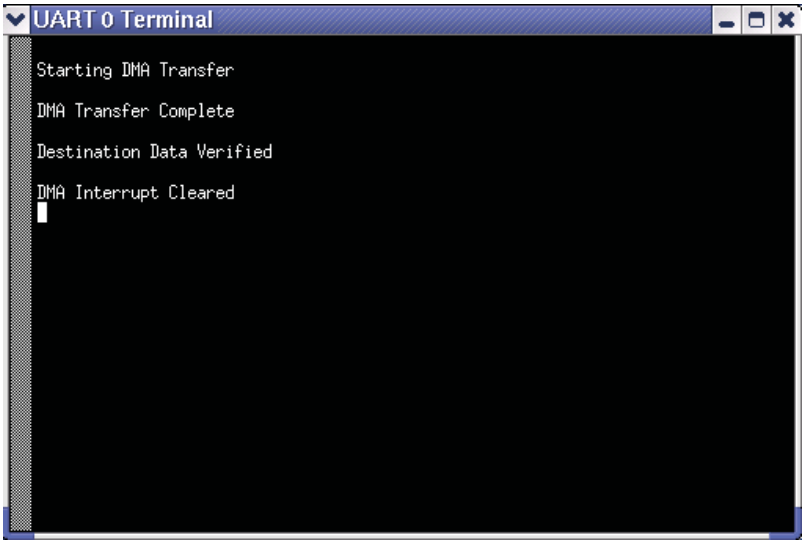


Figure 9: **Hello DMA Output – Linux Simulation Environment**

Simulating the Hello Temac Software Application

This simulation should run for ~ 1250 us. The resulting WLF file is provided at `WLFs/hello_temac.wlf`.

The transfer from HDMA to the XPS LL TEMAC occurs between ~735 us and ~738 us. Watch the dma0ll signals in the processor block.

The XPS LL TEMAC transmits a packet at ~739 us, and the transmit packet from the external transmit pins is looped back to the external receive pins.

The transfer from XPS LL TEMAC to HDMA occurs between ~765 us and ~767 us. Monitor the lldma0 signals in the processor block.

In the Windows simulation environment, the output is seen in the ModelSim transcript window after a transmit and receive of a frame. Some of this output is shown in Figure 10.

```
# UART    0 Transmitted Out Char = 0x52 = "R" @ Time = 838695000.0 ps
# UART    0 Transmitted Out Char = 0x65 = "e" @ Time = 843495000.0 ps
# UART    0 Transmitted Out Char = 0x63 = "c" @ Time = 848295000.0 ps
# UART    0 Transmitted Out Char = 0x65 = "e" @ Time = 853095000.0 ps
# UART    0 Transmitted Out Char = 0x69 = "i" @ Time = 857895000.0 ps
# UART    0 Transmitted Out Char = 0x76 = "v" @ Time = 862695000.0 ps
# UART    0 Transmitted Out Char = 0x65 = "e" @ Time = 867495000.0 ps
# UART    0 Transmitted Out Char = 0x64 = "d" @ Time = 872295000.0 ps
# UART    0 Transmitted Out Char = 0x20 = " " @ Time = 877095000.0 ps
# UART    0 Transmitted Out Char = 0x50 = "P" @ Time = 881895000.0 ps
# UART    0 Transmitted Out Char = 0x61 = "a" @ Time = 886695000.0 ps
# UART    0 Transmitted Out Char = 0x63 = "c" @ Time = 891495000.0 ps
# UART    0 Transmitted Out Char = 0x6b = "k" @ Time = 896295000.0 ps
# UART    0 Transmitted Out Char = 0x65 = "e" @ Time = 901095000.0 ps
# UART    0 Transmitted Out Char = 0x74 = "t" @ Time = 905895000.0 ps
# UART    0 Transmitted Out Char = 0x21 = "!" @ Time = 910695000.0 ps
```

X1003_10_062508

Figure 10: **Hello TEMAC Output – Windows Simulation Environment**

Figure 11 shows the output in the Linux simulation environment.

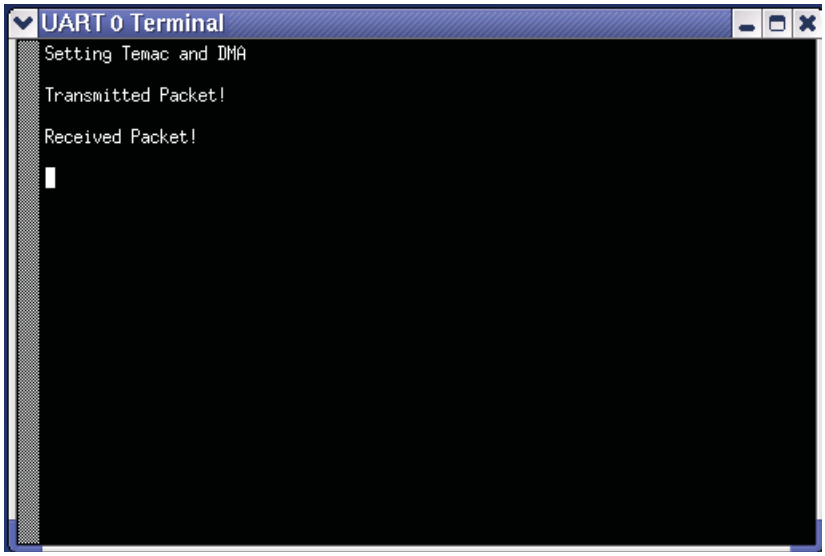


Figure 11: **Hello TEMAC Output – Linux Simulation Environment**

Executing the 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**. See Figure 12 for the settings. This is necessary to see the results from the software application.

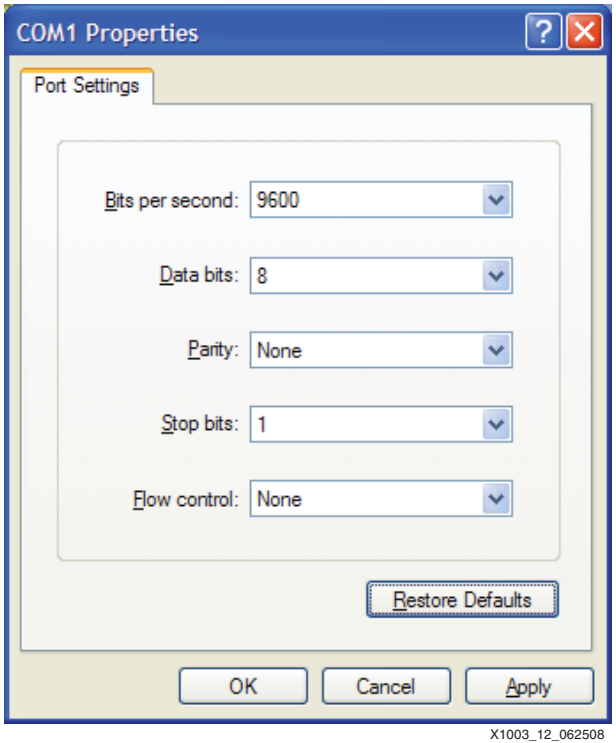


Figure 12: **HyperTerminal Settings**

Executing the 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 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 command:

   ```
   impact -batch xapp1003.cmd
   ```

3. Invoke XMD and connect to the PPC440 processor by using the following command:

   ```
   xmd -opt xapp1003.opt
   ```

4. Download the executables by using the following command depending on the software application:
   - **`dow hello_uart.elf`**
   - **`dow hello_gpio.elf`**
   - **`dow hello_dma.elf`**
   - **`dow hello_temac.elf`**
   - **`dow TestApp_Memory.elf`**
5. Enter in the `run` command to run the software application.

Executing the Reference System from XPS for Hardware

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

1. Open `system.xmp` in XPS.
2. Modify the `system.mhs` under the PPC440MC DDR2 instance from `PARAMETER C_SIM_ONLY = 1` to `PARAMETER C_SIM_ONLY = 0`.
3. Use **Hardware→Generate Bitstream** to generate a bitstream for the system.
4. Click on **Software→Launch Platform Studio SDK**.
5. Once SDK starts, select **Cancel** In the Application Wizard.
6. Select **Project→Build Automatically**. Before proceeding to the next step, ensure that all software applications are built. During this time the software projects and libraries are recompiled.
7. In the SDK project, click on **Device Configuration→Program FPGA**. This will program the board with the PPC440 Bootloop.
8. In the Binaries Tree node of the software application, right click on **<software application>.elf** and select **Run As→1 Run on Hardware**.

Note: Software applications for hardware do not have the _sim extension.

Running the Software Applications

Running the Hello UART Software Application

After running the software application, the output reads as follows:

```
Hello World!
```

Running the Hello GPIO Software Application

Monitor the 8-LEDs on the ML507 board. The LEDs beginning lighting up sequentially from left to right six times. No output is displayed.

Running the TestApp Memory Software Application

The RS232 terminal shows the status of the 32-bit, 16-bit, 8-bit memory tests. The following output should be displayed:

```
Running 32-bit test...PASSED!
Running 16-bit test...PASSED!
Running 8-bit test...PASSED!
```

Running Hello DMA Software Application

The following output should be displayed after the DMA transaction:

```
Starting DMA Transfer
DMA Transfer Complete
Destination Data Verified
DMA Interrupt Cleared
```

Running Hello Temac Software Application

After running the software application, the following output should be displayed:

```
Setting Temac and DMA
Transmitted Packet!
Received Packet!
```

Adding Peripherals to the Initial System

Adding Slave Peripherals

To take advantage of the processor block, slave cores are added to the PLB v4.6 on the MPLB port. In addition, the slave interface on master/slave peripherals are connected to the PLB v4.6 on the MPLB port as well. In this system, the address of the slave must be between `0x40000000` and `0xFFFFFFFF`.

Adding Master Peripherals

As stated in the "Adding Slave Peripherals" section, the slave interface of the master is connected to the PLB v4.6 on the MPLB port. The user has an option of adding the master interface to the PLB v4.6 on either the SPLB0 or MPLB port. Connecting the master to the SPLB0 port allows the master to access either the MIB or MPLB. Connecting the master to the MPLB port allows the master to only access slaves on the MPLB.

Known Issues

The simulation of this reference system requires the use of the Micron memory model included in this system. Simulation with other memory models is not supported. The Micron memory model for Windows is modified for smaller memory than MAX_MEM in ddr2.v. ModelSim crashes when using the default MAX_MEM value in Windows. If ModelSim crashes, modify the `define MAX_SIZE in ddr2.v to a smaller value.

The simulation of the various software applications is to be run as set up and defined in this reference system. All of the software applications supplied with this reference system have been compiled and simulated with all of the various compiler optimization levels with and without global pointer optimization. Any modification of the source code, linker scripts, or system is not supported and may cause the simulation to no longer function.

Structural and timing simulations of this reference system have not been fully verified and are therefore not supported.

References

1. UG200 *Embedded Processor Block in Virtex-5 FPGAs Reference Guide*
2. UG443 *PLB v3.4 and OPB to PLB v4.6 System and Core Migration User Guide*

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
4/8/08	1.0	Initial Xilinx release.
9/2/08	1.1	Updated for EDK 10.1.02.

Notice of Disclaimer