HUNT ENGINEERING
Chestnut Court, Burton Row,
Brent Knoll, Somerset, TA9 4BP, UK
Tel: (+44) (0)1278 760188,
Fax: (+44) (0)1278 760199,
Email: sales@hunteng.co.uk
www.hunteng.co.uk
www.hunt-dsp.com

Connecting the PowerPC Processor to HERON-FPGA9 DDR SDRAM

v1.1 R.Williams 14-02-06

Xilinx Virtex-II Pro FPGAs provide one or more embedded PowerPC processor cores along with the usual array of Virtex-II FPGA technology including logic slices, dedicated multipliers and Block RAM.

The HERON-FPGA9 module from HUNT ENGINEERING uses the Virtex-II Pro and provides two banks of external DDR SDRAM memory. This DDR SDRAM memory is available for use by both the FPGA gates and directly by the PowerPC core.

Each bank of DDR SDRAM memory is directly accessible by the PowerPC core using memory load and store operations. A bank of DDR memory may be used to contain instructions or data, or a combination of both via a single connection to the PLB bus.

The same bank of DDR memory can also be accessed directly by the FPGA gates, making it very easy for FPGA data processing functions to interact with the PowerPC core.

This document discusses how to connect the PowerPC to the DDR memory interface of the FPGA. An example project is provided that demonstrates the PowerPC accessing one half of a bank of memory, while the FPGA gates access the other half.

History

Rev 1.0	First written
Rev 1.1	Updated to work alongside example for the HERON-FPGA12.

Other Relevant Documents

In addition to the Xilinx provided documentation for the Virtex-II Pro, and the software tools that go with it, there is another HUNT ENGINEERING document that we assume you have read before this document. This is the "Getting Started with the Embedded PowerPC" document, also referred to as PowerPC Example A. That document works through the design flow for making a design that uses the PowerPC, and how to compile and load a PowerPC program. These subjects will not be covered again in this document.

The Interfaces of the PowerPC

The PowerPC core in the silicon of the Virtex-II Pro FPGAs offers five interface buses. Conventional processors have an external memory interface bus, and the equivalent to that in the V-II Pro is the Processor Local Bus (PLB). The 64-bit PLB is intended for high performance accesses to peripherals like SDRAM.

From the PLB there is a bus bridge component that offers an On-chip Peripheral Bus (OPB). This 32-bit bus is intended for slower peripherals that can be accessed without stalling the PLB. It is also possible to connect peripherals like SDRAM to this interface, the advantage of that seems to be that it is a simpler interface and consumes less FPGA resource. This almost certainly comes at the price of lower speed.

Both of those interfaces are coupled together, so any accesses on either one, will affect the accesses on the other.

Then there is the Device Control Register (DCR) bus, intended for accessing peripheral control registers. This is not intended for data transfers and does not appear in the processor memory map. Rather it is accessed through a special register and instruction.

The PowerPC core also offers two On Chip memory interfaces, one on the Data side (DSOCM), and the other on the instruction side (ISOCM). Here Block RAMs from the FPGA can be connected as if they were on-chip memory of the processor. These interfaces allow you the designer to customise the amount of on chip instruction and data memory according to your architectural requirements.

The DDR memory interface provided by HUNT ENGINEERING has been designed to connect to the PLB bus of the processor.

The PowerPC core has two separate PLB interfaces, one for instruction and one for data. It is possible to implement two completely independent PLB bus connections. Alternatively, one single PLB bus can be implemented and connected to both the instruction and data PLB ports of the core.

For each bank of external DDR SDRAM memory one PLB connection is provided. Therefore on a module such as the HERON-FPGA9 with two banks of memory, it is possible to implement two separate PLB busses, one for instruction and one for data. The PLB connection of the first bank could be connected to the instruction PLB and the PLB connection of the second bank could be connected to the data PLB.

Alternatively with two banks of DDR memory both PLB connections could be connected to one single system PLB bus.

Connecting DDR Memory to the PowerPC

HUNT ENGINEERING provide a component named 'plb2ddr' that must be used when connecting the PowerPC to the external DDR memory that is fitted to a HERON-FPGA9 module.

The component is provided as a 'peripheral core' that must resides in the 'pcores' directory of the EDK project.

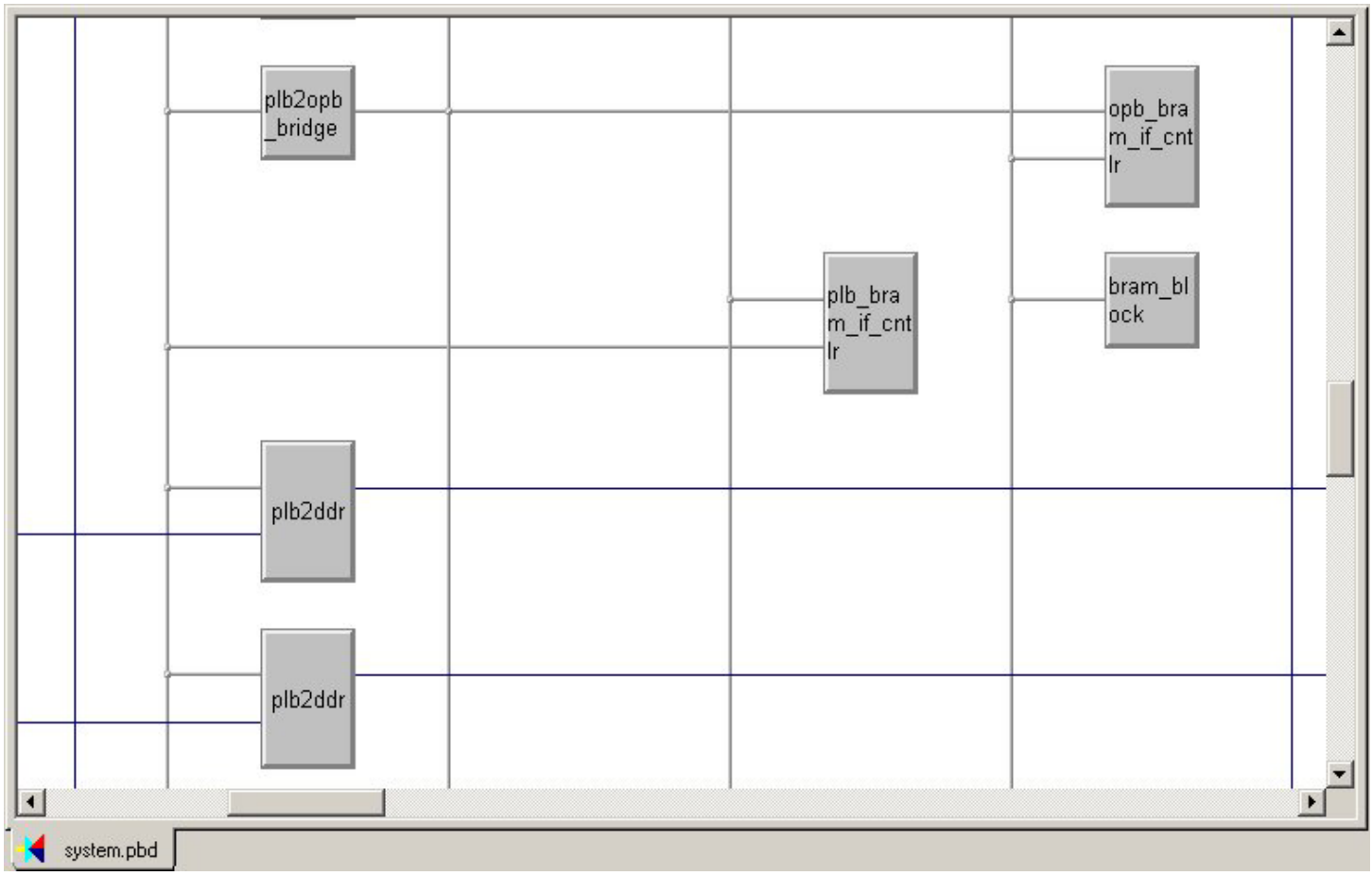
Each placement of the 'plb2ddr' component in the system must correspond to one connection between the chosen PLB bus and one bank of external DDR memory.

Defining the 'PowerPC to DDR' Interfaces

When creating a processor and FPGA design for a Virtex-II Pro HERON-FPGA module both the EDK Platform Studio tool and the Project Navigator tool will be used. Platform Studio provides a means to control and specify how the PowerPC is used and what interfaces are connected to it, and Project Navigator manages the FPGA building process that combines all of the design elements into a single bitstream.

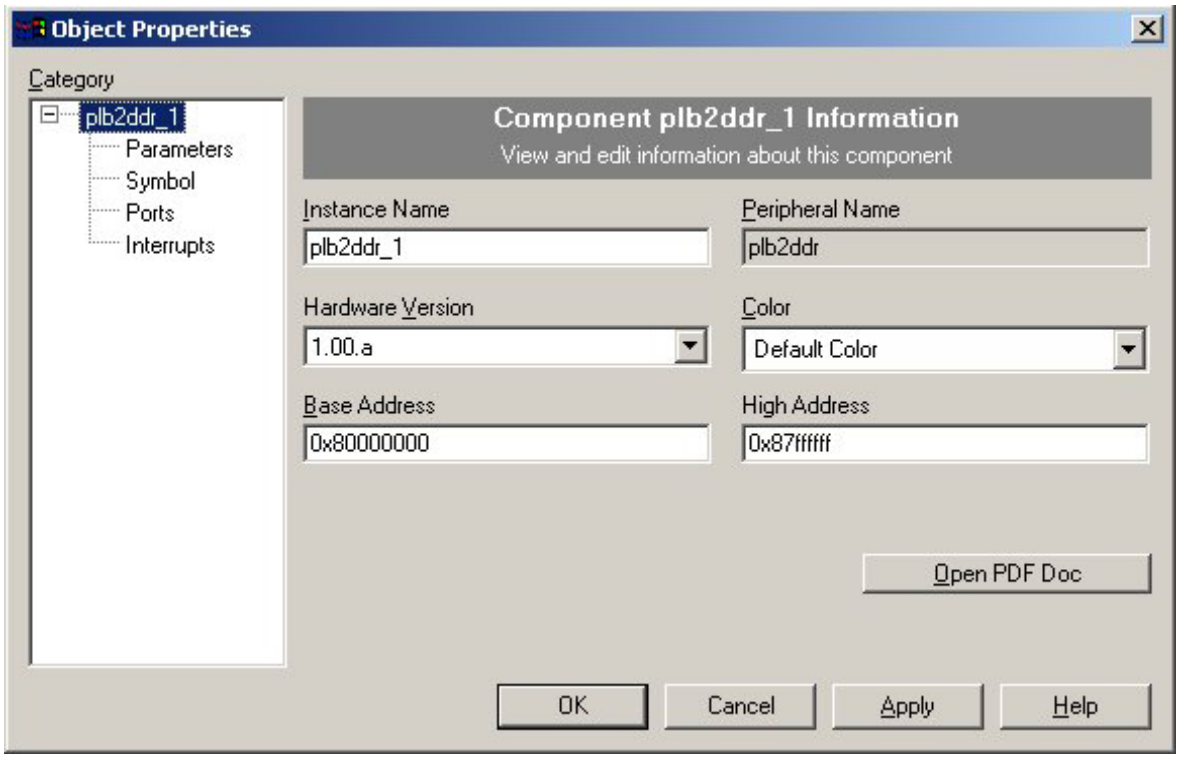
Each bank of DDR memory is placed as one 'plb2ddr' instance in the processor block diagram within the EDK project.

In the example project shown below two banks of DDR memory have been placed on one PLB bus.

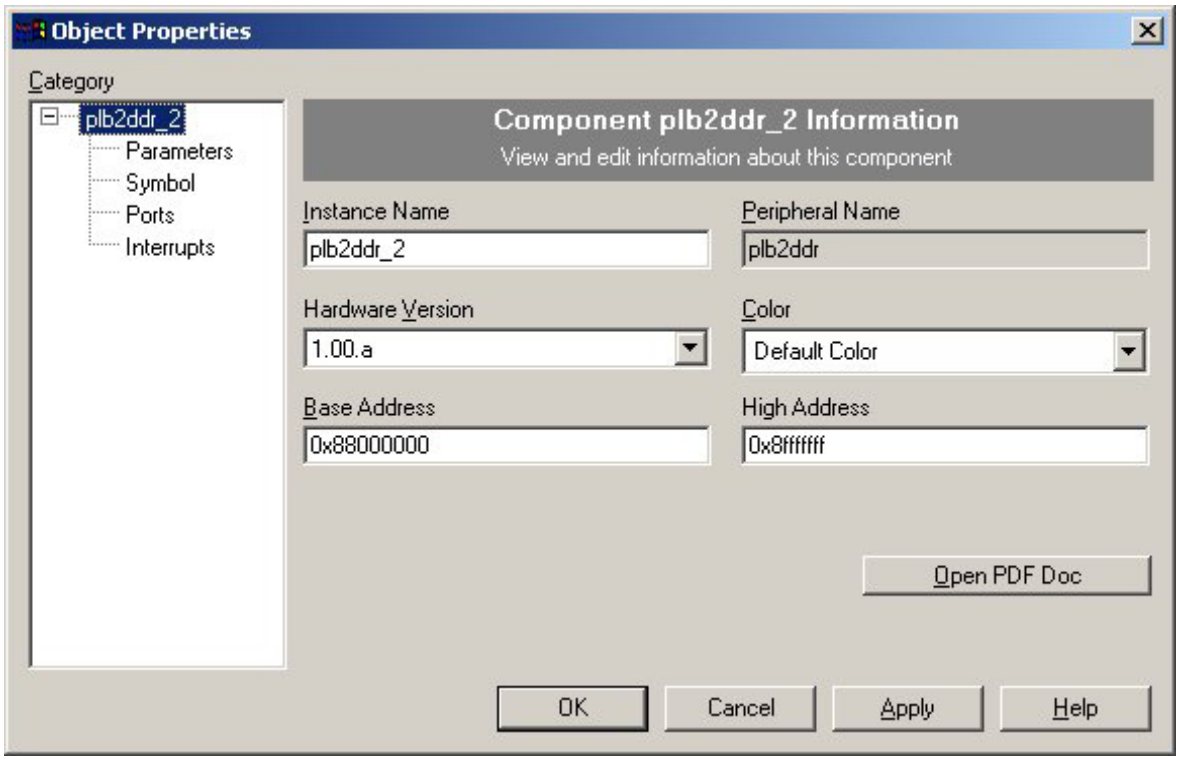


In this block diagram we have connected both the instruction PLB interface and data PLB interface of the processor core to one single PLB bus named 'myplb' to which both banks of DDR are also connected. Therefore, in this project it is possible to store both code and data in the external DDR memory.

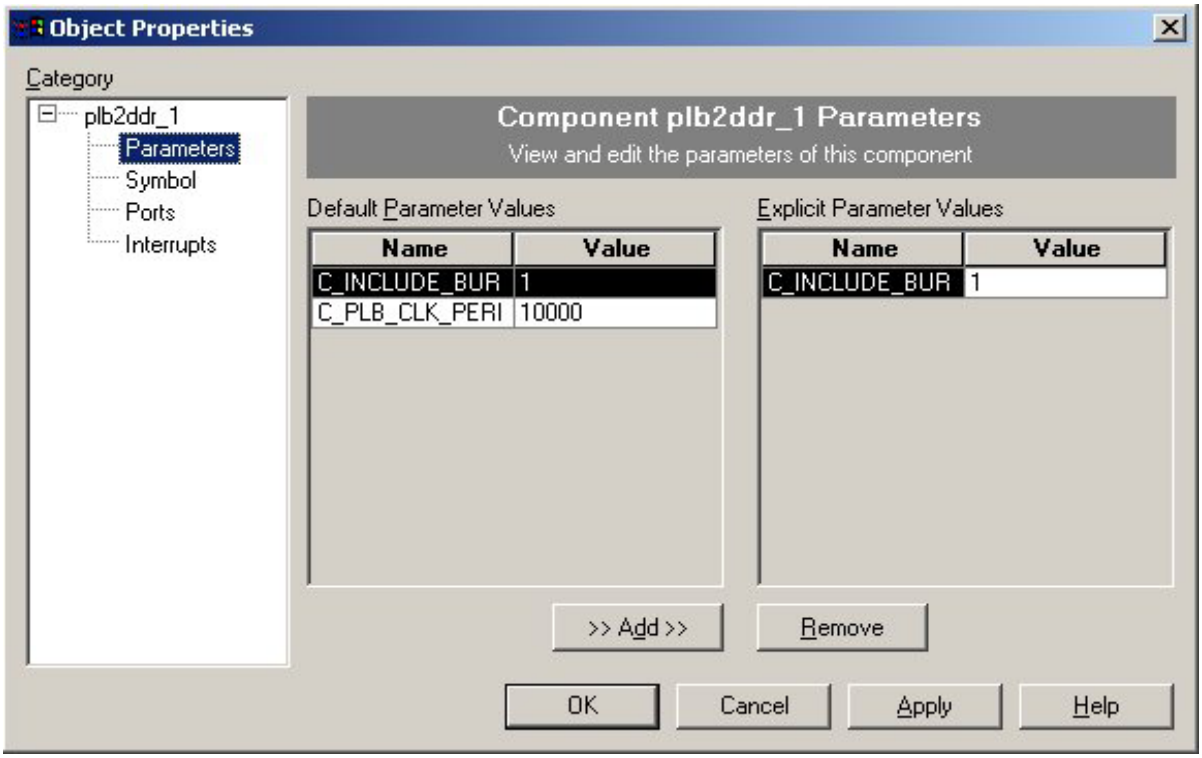
When placing each instance of the 'plb2ddr' component you must correctly specify the address range at which the processor will be able to access the external bank of memory. This address range must be unique within the system and the size of the range must match the amount of memory on that Bank of DDR. The address fields shown below are always set in bytes. In the object properties shown below we can see that the first instance of the 'plb2ddr' component is connected to a 128Mbyte bank of DDR memory starting at address 0x80000000.



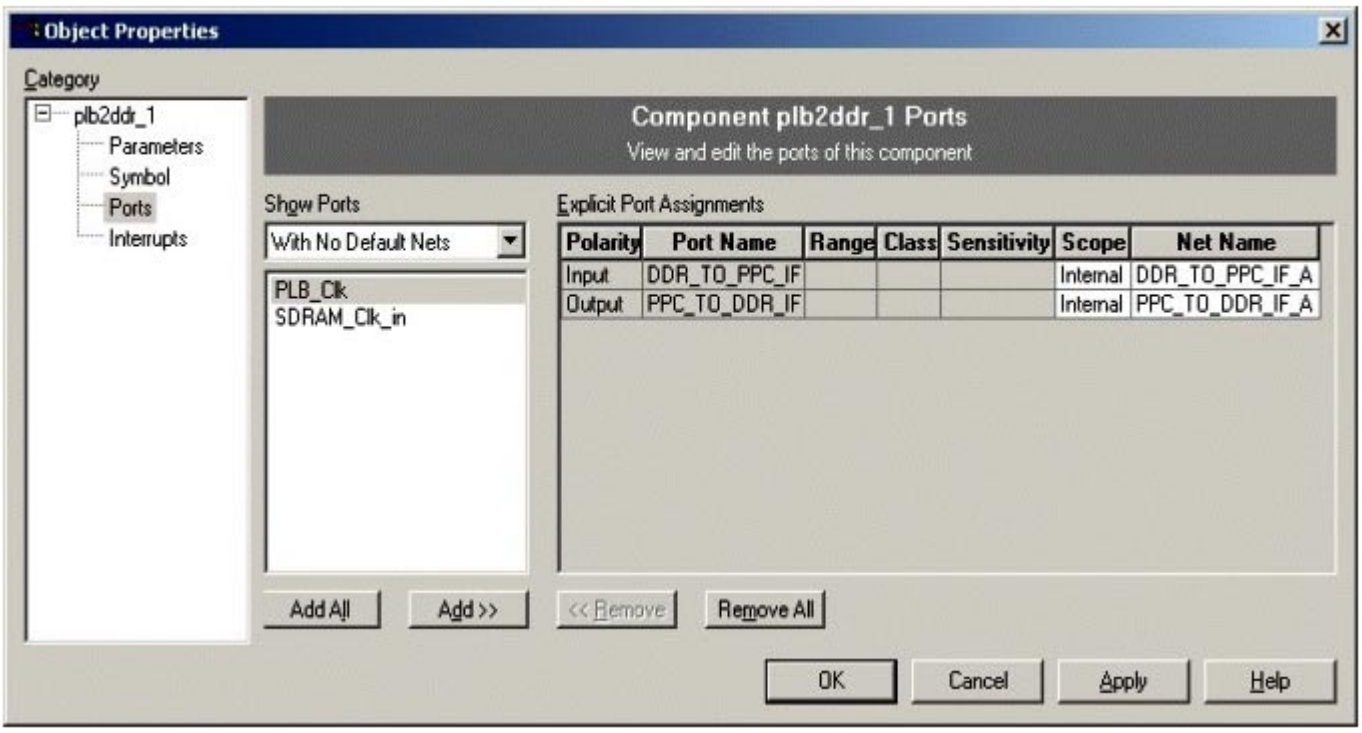
For the object properties of the second instance there is a connection to another 128Mbytes of DDR memory from address 0x88000000. In the example project, both instances combine to give a 256Mbyte DDR memory area from address 0x80000000 to 0x8FFFFFFF.



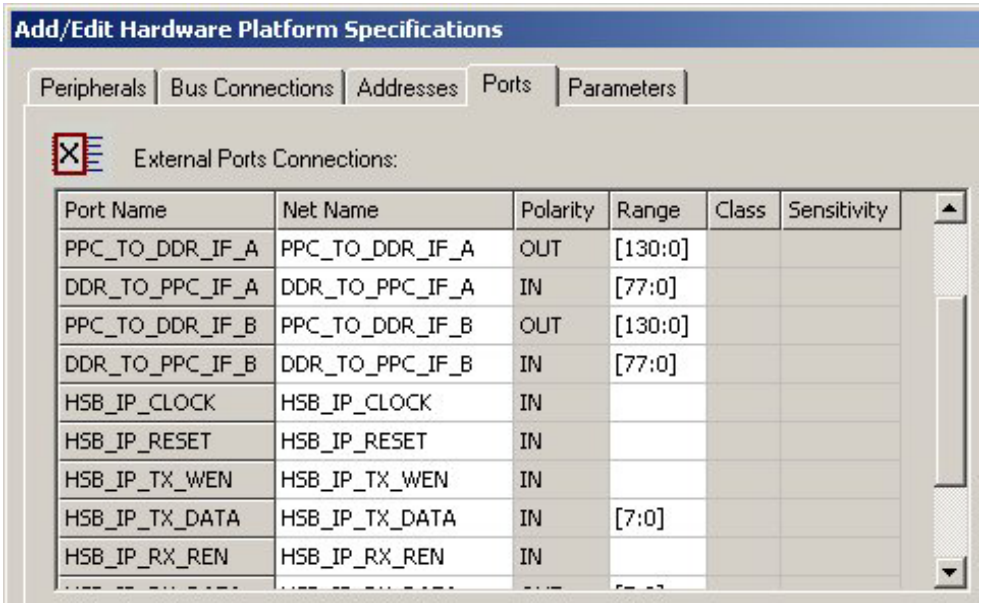
When placing each instance of the 'plb2ddr' component you should ensure that the parameter 'C_INCLUDE_BURST_CACHELN_SUPPORT' is set to 1, to enable burst accesses and cache-line accesses to memory. The 'C_PLB_CLK_PERIOD_PS' setting must be define the number of pico-seconds for the PLB clock period.



When placing each instance of the 'plb2ddr' component, the 'DDR_TO_PPC_IF' port and the 'PPC_TO_DDR_IF' port must be connected to the signals that form the inputs and outputs of the PowerPC project. These signals will then be connected to the appropriate ports of the User-Ap entity in the Project Navigator project.



The picture below shows the external ports of the example project. There are two separate instances of the 'plb2ddr' component. The first 'plb2ddr' instance is connected to the external ports 'PPC_TO_DDR_IF_A' and 'DDR_TO_PPC_IF_A'. The second instance is connected to the external ports 'PPC_TO_DDR_IF_B' and 'DDR_TO_PPC_IF_B'.



Port Name	Net Name	Polarity	Range	Class	Sensitivity
PPC_TO_DDR_IF_A	PPC_TO_DDR_IF_A	OUT	[130:0]		
DDR_TO_PPC_IF_A	DDR_TO_PPC_IF_A	IN	[77:0]		
PPC_TO_DDR_IF_B	PPC_TO_DDR_IF_B	OUT	[130:0]		
DDR_TO_PPC_IF_B	DDR_TO_PPC_IF_B	IN	[77:0]		
HSB_IP_CLOCK	HSB_IP_CLOCK	IN			
HSB_IP_RESET	HSB_IP_RESET	IN			
HSB_IP_TX_WEN	HSB_IP_TX_WEN	IN			
HSB_IP_TX_DATA	HSB_IP_TX_DATA	IN	[7:0]		
HSB_IP_RX_REN	HSB_IP_RX_REN	IN			

Connecting to the USER-AP Entity

The PowerPC project created in the EDK forms a sub-level of the hierarchy in the Project Navigator project. The top-level of the EDK design is usually placed directly in the user-ap level of the Project Navigator project and an example component definition is shown below.

```
component system
  port (
    sys_rst_n        : in  std_logic;
    sys_clk          : in  std_logic;
    JTAG_TRST        : in  std_logic;
    JTAG_HALT        : in  std_logic;
    PPC_TO_DDR_IF_A  : out std_logic_vector(130 downto 0);
    DDR_TO_PPC_IF_A  : in  std_logic_vector(77 downto 0);
    PPC_TO_DDR_IF_B  : out std_logic_vector(130 downto 0);
    DDR_TO_PPC_IF_B  : in  std_logic_vector(77 downto 0);
    HSB_IP_CLOCK     : in  std_logic;
    HSB_IP_RESET     : in  std_logic;
    HSB_IP_TX_WEN    : in  std_logic;
    HSB_IP_TX_DATA   : in  std_logic_vector(7 downto 0);
    HSB_IP_RX_REN    : in  std_logic;
    HSB_IP_RX_DATA   : out std_logic_vector(7 downto 0);
    HSB_IP_COUNT     : out std_logic_vector(7 downto 0);
    HSB_IP_FLAGS     : out std_logic_vector(7 downto 0)
  );
end component;
```

The 'system' component would then be connected as shown below:

```
  isystem : system
    port map (
      sys_rst_n       => PPC_RESET,
      sys_clk         => OSC3,
      JTAG_TRST       => VCC,
      JTAG_HALT       => VCC,
      PPC_TO_DDR_IF_A => PPC_TO_DDR_IF_A,
      DDR_TO_PPC_IF_A => DDR_TO_PPC_IF_A,
      PPC_TO_DDR_IF_B => PPC_TO_DDR_IF_B,
      DDR_TO_PPC_IF_B => DDR_TO_PPC_IF_B,
      HSB_IP_CLOCK    => HSB_IP_CLOCK,
      HSB_IP_RESET    => HSB_IP_RESET,
      HSB_IP_TX_WEN   => HSB_IP_TX_WEN,
      HSB_IP_TX_DATA  => HSB_IP_TX_DATA,
      HSB_IP_RX_REN   => HSB_IP_RX_REN,
      HSB_IP_RX_DATA  => HSB_IP_RX_DATA,
      HSB_IP_COUNT    => HSB_IP_COUNT,
      HSB_IP_FLAGS    => HSB_IP_FLAGS );
```

When connecting to the ports of the 'system' component, the signals '`PPC_TO_DDR_IF_A`', '`DDR_TO_PPC_IF_A`', '`PPC_TO_DDR_IF_B`' and '`DDR_TO_PPC_IF_B`' must be directly connected to the identically named ports of the 'User-Ap' entity for the module type you are using.

For example, the last few lines of the 'User-Ap' entity of the HERON-FPGA9 are shown below:

```
    DDR_B_RD_RISE_AE  : in  std_logic;
    DDR_B_RD_FALL_REN : out std_logic;
    DDR_B_RD_FALL_EF  : in  std_logic;
    DDR_B_RD_FALL_AE  : in  std_logic;
-- DDR PPC Interface for Port B
    PPC_TO_DDR_IF_A   : out std_logic_vector(130 downto 0);
    DDR_TO_PPC_IF_A   : in  std_logic_vector(77 downto 0);
-- DDR PPC Interface for Port B
    PPC_TO_DDR_IF_B   : out std_logic_vector(130 downto 0);
    DDR_TO_PPC_IF_B   : in  std_logic_vector(77 downto 0)
  );
end USER_AP;
```

The Example Project

In the PowerPC to DDR Example Project the FPGA gates implement very similar logic to the standard HUNT ENGINEERING memory test project (Example2). The FPGA gates in this example are used to read and write the first 64Mbytes of DDR memory Bank A.

The PowerPC is also connected to Bank A and a program is provided that performs a read-write memory test of the top 64Mbytes of memory.

The PowerPC design includes a component on the OPB bus that is used to communicating with the HSB logic of the FPGA. This component enables the PowerPC program to communicate with the host PC based example program.

The host program controls the reading and writing of the memory test performed by the FPGA gates. It also expects HSB messages from the PowerPC that keeps it informed of the loop number, error count, write speed and read speed.

The PowerPC Program

The PowerPC program operates in a continuous loop. For each iteration of the loop it performs five memory tests to the DDR memory between byte address 0x84000000 to 0x88000000. This address range corresponds to the top half of Bank A.

The five tests performed are writing all bits set to 0, reading back and comparing, followed by the pattern of all bits set to 1, the value 0x5, then the value 0xA, and finally writing and reading with the address.

After these five tests are completed six 32-bit words are sent to the host via the OPB2HSB component. The first word sent contains the current PowerPC loop number. The second word contains the error count. The third and fourth words contain a 64-bit timer value for write performance calculation. The fifth and sixth words contain a 64-bit timer value for read performance calculations.

The Host Program

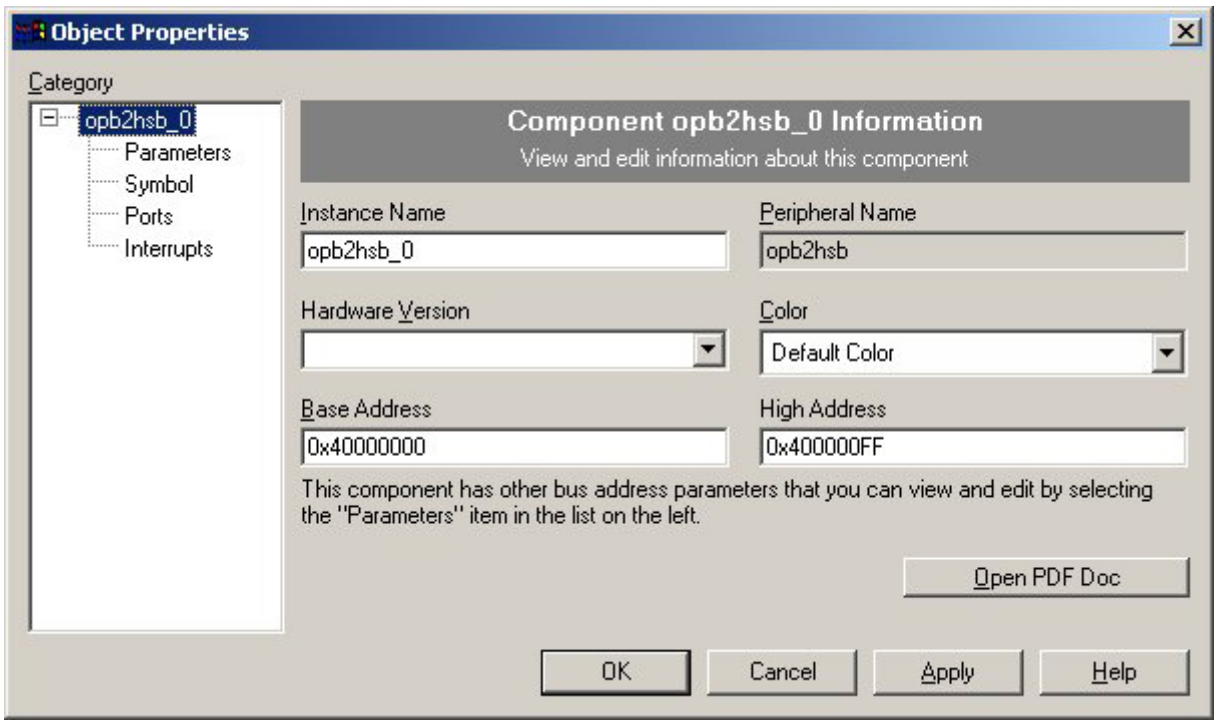
The Host program is very similar to the standard memory test routine supplied with Example 2. In the PowerPC to DDR example, only the first half of each memory bank is tested by the FPGA gates. At the end of each memory test loop, the Host program expects to receive six words from the PowerPC program. The words that are received are used for displaying the PowerPC loop count, error count, write memory access speed and read memory access speed.

The OPB 2 HSB Module

The 'opb2hsb' module is provided as part of the PowerPC to DDR example for communication between the host program and PowerPC.

The 'opb2hsb' module connects to the OPB interface of the PowerPC. In order to use OPB in your system design, a PLB connection must be directly made to the processor core and a PLB2OPB bridge must be used to connect to an OPB bus.

In the example project, the 'opb2hsb' module is placed in the address range of 0x40000000 to 0x400000FF. The module only needs a very small address range that is at least 16 bytes (4 words) in size. The example uses a slightly larger range to reduce the address decoding logic that is generated.



The address map of the 'opb2hsb' module is shown below. Please be aware, the address offsets are specified in 32-bit word steps, while the Example Address reflects the physical byte address for each function.

Word Address Offset	Example Address	Function
0x00	0x40000000	OPB2HSB Data
0x01	0x40000004	OPB2HSB Count Register
0x02	0x40000008	OPB2HSB Flags Register

The OPB2HSB implements a bi-directional byte-wide FIFO between OPB and the HSB interface controlled by the FPGA gates.

At address offset 0, data is either read from the inbound side of the FIFO (from HSB) side or is written to the outbound side of the FIFO (to HSB).

At address offset 1 there is an 8-bit register that indicates how many bytes can be sent in the bottom four bits, and how many bytes can be read in the top four bits.

At address offset 2 there is an 8-bit flags register that indicates the outbound FIFO flag conditions in the bottom four bits and the inbound FIFO flag conditions in the top four bits.

In the example program, a function is provided that manages the use of the OPB2HSB Count Register in order to safely control the sending of words to the Host program.

Building the Example Project

In the root directory of the PowerPC to DDR Example are the project files required by the EDK. In the 'pcores' sub-directory there are three peripheral cores.

The first peripheral core is the 'plb2ddr' component. When making a new project that accesses DDR SDRAM, you will need to copy the entire 'plb2ddr' component tree from the example 'pcores' directory to your own 'pcores' directory.

The second peripheral core is the 'opb2hsb' component explained above.

The third and final peripheral is the 'system-clock' component, which contains a DCM for controlling the CPU clock at 200MHz and PLB clock at 100MHz.

In addition to the 'pcores' project sub-directory, there is an 'ISE' sub-directory that contains all of the Project Navigator project files, and a 'Src' sub-directory that contains all of the VHDL source used by the Project Navigator project.

In order to create a bitstream for the PowerPC to DDR example, a net-list must first be created in the EDK Platform Studio. To generate the PowerPC system net-list open Platform Studio and then open the project file '`system.xmp`'.

With this done select 'Tools→Generate Netlist' in Platform Studio. When the net-list generation has completed successfully the PowerPC design must be exported to Project Navigator. Please note, this MUST NOT be done using the menu item 'Tools→Export to ProjNav'. Instead, you must use the supplied batch file '`Export_to_ProjNav.bat`' supplied in the root example directory.

When the export process has successfully completed open the ISE project in Project Navigator. To build the bitstream select the file 'top.vhd' in the Module View and double click on Generate Programming File in the Process View.

When the bitstream has been generated in Project Navigator you will need to return to Platform Studio to build the PowerPC program and combine the resulting executable with the bitstream so that the Block RAMs contain the correct pre-initialised data.

First select the menu item 'Tools→Import from ProjNav'.

Build the C program by selecting the menu item 'Tools→Build All User Applications'. When this has completed combine the executable with the previously generated bitstream using 'Tools→Update Bitstream'. The final bitstream 'download.bit', can now be downloaded to the target FPGA. This file is written to the implementation directory by Platform Studio.

DDR Memory Performance

The 'plb2ddr' component has been designed to respond to both single memory load and store operations, multiple word load and store operations, and cache-line burst reads and cache-line burst writes.

The performance that can be achieved between the processor core and external memory will very much depend on the rate at which the processor can generate data accesses. The FPGA logic that interfaces the external memory to the processor is capable of transfer 64-bits of data on every PLB clock cycle during a memory burst (cache or load/store multiple) access. With a typical PLB clock rate of 100MHz this equates to a burst rate of 800Mbytes/sec.

When performing single word accesses however, this data rate will drop significantly. For each data word transferred over PLB there will be a large amount of overhead for the clock cycles where data is not being transferred. The kind of performance that can be expected in the single word access mode can be seen by running the example host program with the example bitstream downloaded to your Virtex-II Pro based FPGA module.

For each stage of the PowerPC memory test sequence a count is kept of the amount of time taken to perform all write accesses and the time taken to perform all read accesses. These numbers are transferred via HSB to the host program where the memory bandwidth is calculated.

In the PowerPC example program, the instruction cache is enabled at the start of operation to increase the rate at which the processor is able to generate accesses to DDR memory.

When running the example program, you should notice that the memory write bandwidth is in the order of 57Mbytes/sec while the memory read bandwidth is in the order of 19Mbytes/sec. The reason for this is that when a read is issued in single access mode, the PLB bus cannot generate another read access until the data has arrived from the first access. Therefore, the read request must be processed by the FPGA gates, an access generated to the external DDR memory, the data received and then passed back to the processor. In the case of memory writes however, the DDR memory interface allows the writes to be pipelined such that the PLB bus is free to generate the next data word while the last access is still in progress.