





























## C:\Xilinx\11.1\edk\_user\_repository\MyProcessorIPLib\pcores\plb\_user\_logic\_16x8192\_5cs\_10clk\_v1\_00\_a\data

------------------------------------------------------------------------------

-- user\_logic.vhd - entity/architecture pair

------------------------------------------------------------------------------

--

-- \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

-- \*\* Copyright (c) 1995-2009 Xilinx, Inc. All rights reserved. \*\*

-- \*\* \*\*

-- \*\* Xilinx, Inc. \*\*

-- \*\* XILINX IS PROVIDING THIS DESIGN, CODE, OR INFORMATION "AS IS" \*\*

-- \*\* AS A COURTESY TO YOU, SOLELY FOR USE IN DEVELOPING PROGRAMS AND \*\*

-- \*\* SOLUTIONS FOR XILINX DEVICES. BY PROVIDING THIS DESIGN, CODE, \*\*

-- \*\* OR INFORMATION AS ONE POSSIBLE IMPLEMENTATION OF THIS FEATURE, \*\*

-- \*\* APPLICATION OR STANDARD, XILINX IS MAKING NO REPRESENTATION \*\*

-- \*\* THAT THIS IMPLEMENTATION IS FREE FROM ANY CLAIMS OF INFRINGEMENT, \*\*

-- \*\* AND YOU ARE RESPONSIBLE FOR OBTAINING ANY RIGHTS YOU MAY REQUIRE \*\*

-- \*\* FOR YOUR IMPLEMENTATION. XILINX EXPRESSLY DISCLAIMS ANY \*\*

-- \*\* WARRANTY WHATSOEVER WITH RESPECT TO THE ADEQUACY OF THE \*\*

-- \*\* IMPLEMENTATION, INCLUDING BUT NOT LIMITED TO ANY WARRANTIES OR \*\*

-- \*\* REPRESENTATIONS THAT THIS IMPLEMENTATION IS FREE FROM CLAIMS OF \*\*

-- \*\* INFRINGEMENT, IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS \*\*

-- \*\* FOR A PARTICULAR PURPOSE. \*\*

-- \*\* \*\*

-- \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

--

------------------------------------------------------------------------------

-- Filename: user\_logic.vhd

-- Version: 1.00.a

-- Description: User logic.

-- Date: Tue Apr 06 13:46:31 2010 (by Create and Import Peripheral Wizard)

-- VHDL Standard: VHDL'93

------------------------------------------------------------------------------

-- Naming Conventions:

-- active low signals: "\*\_n"

-- clock signals: "clk", "clk\_div#", "clk\_#x"

-- reset signals: "rst", "rst\_n"

-- generics: "C\_\*"

-- user defined types: "\*\_TYPE"

-- state machine next state: "\*\_ns"

-- state machine current state: "\*\_cs"

-- combinatorial signals: "\*\_com"

-- pipelined or register delay signals: "\*\_d#"

-- counter signals: "\*cnt\*"

-- clock enable signals: "\*\_ce"

-- internal version of output port: "\*\_i"

-- device pins: "\*\_pin"

-- ports: "- Names begin with Uppercase"

-- processes: "\*\_PROCESS"

-- component instantiations: "<ENTITY\_>I\_<#|FUNC>"

------------------------------------------------------------------------------

-- DO NOT EDIT BELOW THIS LINE --------------------

library ieee;

use ieee.std\_logic\_1164.all;

use ieee.std\_logic\_arith.all;

use ieee.std\_logic\_unsigned.all;

library proc\_common\_v3\_00\_a;

use proc\_common\_v3\_00\_a.proc\_common\_pkg.all;

-- DO NOT EDIT ABOVE THIS LINE --------------------

--USER libraries added here

------------------------------------------------------------------------------

-- Entity section

------------------------------------------------------------------------------

-- Definition of Generics:

-- C\_SLV\_AWIDTH -- Slave interface address bus width

-- C\_SLV\_DWIDTH -- Slave interface data bus width

-- C\_NUM\_MEM -- Number of memory spaces

--

-- Definition of Ports:

-- Bus2IP\_Clk -- Bus to IP clock

-- Bus2IP\_Reset -- Bus to IP reset

-- Bus2IP\_Addr -- Bus to IP address bus

-- Bus2IP\_CS -- Bus to IP chip select for user logic memory selection

-- Bus2IP\_RNW -- Bus to IP read/not write

-- Bus2IP\_Data -- Bus to IP data bus

-- Bus2IP\_BE -- Bus to IP byte enables

-- IP2Bus\_Data -- IP to Bus data bus

-- IP2Bus\_RdAck -- IP to Bus read transfer acknowledgement

-- IP2Bus\_WrAck -- IP to Bus write transfer acknowledgement

-- IP2Bus\_Error -- IP to Bus error response

------------------------------------------------------------------------------

entity user\_logic is

generic

(

-- ADD USER GENERICS BELOW THIS LINE ---------------

--USER generics added here

-- ADD USER GENERICS ABOVE THIS LINE ---------------

-- DO NOT EDIT BELOW THIS LINE ---------------------

-- Bus protocol parameters, do not add to or delete

C\_SLV\_AWIDTH : integer := 32;

C\_SLV\_DWIDTH : integer := 32;

C\_NUM\_MEM : integer := 1

-- DO NOT EDIT ABOVE THIS LINE ---------------------

);

port

(

-- ADD USER PORTS BELOW THIS LINE ------------------

--USER ports added here

-- ADD USER PORTS ABOVE THIS LINE ------------------

-- DO NOT EDIT BELOW THIS LINE ---------------------

-- Bus protocol ports, do not add to or delete

Bus2IP\_Clk : in std\_logic;

Bus2IP\_Reset : in std\_logic;

Bus2IP\_Addr : in std\_logic\_vector(0 to C\_SLV\_AWIDTH-1);

Bus2IP\_CS : in std\_logic\_vector(0 to C\_NUM\_MEM-1);

Bus2IP\_RNW : in std\_logic;

Bus2IP\_Data : in std\_logic\_vector(0 to C\_SLV\_DWIDTH-1);

Bus2IP\_BE : in std\_logic\_vector(0 to C\_SLV\_DWIDTH/8-1);

IP2Bus\_Data : out std\_logic\_vector(0 to C\_SLV\_DWIDTH-1);

IP2Bus\_RdAck : out std\_logic;

IP2Bus\_WrAck : out std\_logic;

IP2Bus\_Error : out std\_logic

-- DO NOT EDIT ABOVE THIS LINE ---------------------

);

attribute SIGIS : string;

attribute SIGIS of Bus2IP\_Clk : signal is "CLK";

attribute SIGIS of Bus2IP\_Reset : signal is "RST";

end entity user\_logic;

------------------------------------------------------------------------------

-- Architecture section

------------------------------------------------------------------------------

architecture IMP of user\_logic is

--USER signal declarations added here, as needed for user logic

------------------------------------------

-- Signals for user logic memory space example

------------------------------------------

type BYTE\_RAM\_TYPE is array (0 to 255) of std\_logic\_vector(0 to 7);

type DO\_TYPE is array (0 to C\_NUM\_MEM-1) of std\_logic\_vector(0 to C\_SLV\_DWIDTH-1);

signal mem\_data\_out : DO\_TYPE;

signal mem\_address : std\_logic\_vector(0 to 7);

signal mem\_select : std\_logic\_vector(0 to 0);

signal mem\_read\_enable : std\_logic;

signal mem\_read\_enable\_dly1 : std\_logic;

signal mem\_read\_req : std\_logic;

signal mem\_ip2bus\_data : std\_logic\_vector(0 to C\_SLV\_DWIDTH-1);

signal mem\_read\_ack\_dly1 : std\_logic;

signal mem\_read\_ack : std\_logic;

signal mem\_write\_ack : std\_logic;

begin

--USER logic implementation added here

------------------------------------------

-- Example code to access user logic memory region

--

-- Note:

-- The example code presented here is to show you one way of using

-- the user logic memory space features. The Bus2IP\_Addr, Bus2IP\_CS,

-- and Bus2IP\_RNW IPIC signals are dedicated to these user logic

-- memory spaces. Each user logic memory space has its own address

-- range and is allocated one bit on the Bus2IP\_CS signal to indicated

-- selection of that memory space. Typically these user logic memory

-- spaces are used to implement memory controller type cores, but it

-- can also be used in cores that need to access additional address space

-- (non C\_BASEADDR based), s.t. bridges. This code snippet infers

-- 1 256x32-bit (byte accessible) single-port Block RAM by XST.

------------------------------------------

mem\_select <= Bus2IP\_CS;

mem\_read\_enable <= ( Bus2IP\_CS(0) ) and Bus2IP\_RNW;

mem\_read\_ack <= mem\_read\_ack\_dly1;

mem\_write\_ack <= ( Bus2IP\_CS(0) ) and not(Bus2IP\_RNW);

mem\_address <= Bus2IP\_Addr(C\_SLV\_AWIDTH-10 to C\_SLV\_AWIDTH-3);

-- implement single clock wide read request

mem\_read\_req <= mem\_read\_enable and not(mem\_read\_enable\_dly1);

BRAM\_RD\_REQ\_PROC : process( Bus2IP\_Clk ) is

begin

if ( Bus2IP\_Clk'event and Bus2IP\_Clk = '1' ) then

if ( Bus2IP\_Reset = '1' ) then

mem\_read\_enable\_dly1 <= '0';

else

mem\_read\_enable\_dly1 <= mem\_read\_enable;

end if;

end if;

end process BRAM\_RD\_REQ\_PROC;

-- this process generates the read acknowledge 1 clock after read enable

-- is presented to the BRAM block. The BRAM block has a 1 clock delay

-- from read enable to data out.

BRAM\_RD\_ACK\_PROC : process( Bus2IP\_Clk ) is

begin

if ( Bus2IP\_Clk'event and Bus2IP\_Clk = '1' ) then

if ( Bus2IP\_Reset = '1' ) then

mem\_read\_ack\_dly1 <= '0';

else

mem\_read\_ack\_dly1 <= mem\_read\_req;

end if;

end if;

end process BRAM\_RD\_ACK\_PROC;

-- implement Block RAM(s)

BRAM\_GEN : for i in 0 to C\_NUM\_MEM-1 generate

constant NUM\_BYTE\_LANES : integer := (C\_SLV\_DWIDTH+7)/8;

begin

BYTE\_BRAM\_GEN : for byte\_index in 0 to NUM\_BYTE\_LANES-1 generate

signal ram : BYTE\_RAM\_TYPE;

signal write\_enable : std\_logic;

signal data\_in : std\_logic\_vector(0 to 7);

signal data\_out : std\_logic\_vector(0 to 7);

signal read\_address : std\_logic\_vector(0 to 7);

begin

write\_enable <= not(Bus2IP\_RNW) and

Bus2IP\_CS(i) and

Bus2IP\_BE(byte\_index);

data\_in <= Bus2IP\_Data(byte\_index\*8 to byte\_index\*8+7);

mem\_data\_out(i)(byte\_index\*8 to byte\_index\*8+7) <= data\_out;

BYTE\_RAM\_PROC : process( Bus2IP\_Clk ) is

begin

if ( Bus2IP\_Clk'event and Bus2IP\_Clk = '1' ) then

if ( write\_enable = '1' ) then

ram(CONV\_INTEGER(mem\_address)) <= data\_in;

end if;

read\_address <= mem\_address;

end if;

end process BYTE\_RAM\_PROC;

data\_out <= ram(CONV\_INTEGER(read\_address));

end generate BYTE\_BRAM\_GEN;

end generate BRAM\_GEN;

-- implement Block RAM read mux

MEM\_IP2BUS\_DATA\_PROC : process( mem\_data\_out, mem\_select ) is

begin

case mem\_select is

when "1" => mem\_ip2bus\_data <= mem\_data\_out(0);

when others => mem\_ip2bus\_data <= (others => '0');

end case;

end process MEM\_IP2BUS\_DATA\_PROC;

------------------------------------------

-- Example code to drive IP to Bus signals

------------------------------------------

IP2Bus\_Data <= mem\_ip2bus\_data when mem\_read\_ack = '1' else

(others => '0');

IP2Bus\_WrAck <= mem\_write\_ack;

IP2Bus\_RdAck <= mem\_read\_ack;

IP2Bus\_Error <= '0';

end IMP;

###################################################################

##

## Name : plb\_user\_logic\_16x8192\_5cs\_10clk

## Desc : Microprocessor Peripheral Description

## : Automatically generated by PsfUtility

##

###################################################################

BEGIN plb\_user\_logic\_16x8192\_5cs\_10clk

## Peripheral Options

OPTION IPTYPE = PERIPHERAL

OPTION IMP\_NETLIST = TRUE

OPTION HDL = VHDL

OPTION IP\_GROUP = MICROBLAZE:PPC:USER

OPTION DESC = PLB\_USER\_LOGIC\_16X8192\_5CS\_10CLK

## Bus Interfaces

BUS\_INTERFACE BUS = SPLB, BUS\_STD = PLBV46, BUS\_TYPE = SLAVE

## Generics for VHDL or Parameters for Verilog

PARAMETER C\_SPLB\_AWIDTH = 32, DT = INTEGER, BUS = SPLB, ASSIGNMENT = CONSTANT

PARAMETER C\_SPLB\_DWIDTH = 128, DT = INTEGER, BUS = SPLB, RANGE = (32, 64, 128)

PARAMETER C\_SPLB\_NUM\_MASTERS = 8, DT = INTEGER, BUS = SPLB, RANGE = (1:16)

PARAMETER C\_SPLB\_MID\_WIDTH = 3, DT = INTEGER, BUS = SPLB, RANGE = (1:4)

PARAMETER C\_SPLB\_NATIVE\_DWIDTH = 32, DT = INTEGER, BUS = SPLB, RANGE = (32, 64, 128), ASSIGNMENT = CONSTANT

PARAMETER C\_SPLB\_P2P = 0, DT = INTEGER, BUS = SPLB, RANGE = (0, 1)

PARAMETER C\_SPLB\_SUPPORT\_BURSTS = 0, DT = INTEGER, BUS = SPLB, RANGE = (0, 1), ASSIGNMENT = CONSTANT

PARAMETER C\_SPLB\_SMALLEST\_MASTER = 32, DT = INTEGER, BUS = SPLB, RANGE = (32, 64, 128)

PARAMETER C\_SPLB\_CLK\_PERIOD\_PS = 10000, DT = INTEGER, BUS = SPLB

PARAMETER C\_INCLUDE\_DPHASE\_TIMER = 0, DT = INTEGER, RANGE = (0, 1)

PARAMETER C\_FAMILY = virtex5, DT = STRING

PARAMETER C\_MEM0\_BASEADDR = 0xffffffff, DT = std\_logic\_vector, PAIR = C\_MEM0\_HIGHADDR, ADDRESS = BASE, BUS = SPLB

PARAMETER C\_MEM0\_HIGHADDR = 0x00000000, DT = std\_logic\_vector, PAIR = C\_MEM0\_BASEADDR, ADDRESS = HIGH, BUS = SPLB

PARAMETER C\_MEM1\_BASEADDR = 0xffffffff, DT = std\_logic\_vector, PAIR = C\_MEM1\_HIGHADDR, ADDRESS = BASE, BUS = SPLB

PARAMETER C\_MEM1\_HIGHADDR = 0x00000000, DT = std\_logic\_vector, PAIR = C\_MEM1\_BASEADDR, ADDRESS = HIGH, BUS = SPLB

PARAMETER C\_MEM2\_BASEADDR = 0xffffffff, DT = std\_logic\_vector, PAIR = C\_MEM2\_HIGHADDR, ADDRESS = BASE, BUS = SPLB

PARAMETER C\_MEM2\_HIGHADDR = 0x00000000, DT = std\_logic\_vector, PAIR = C\_MEM2\_BASEADDR, ADDRESS = HIGH, BUS = SPLB

PARAMETER C\_MEM3\_BASEADDR = 0xffffffff, DT = std\_logic\_vector, PAIR = C\_MEM3\_HIGHADDR, ADDRESS = BASE, BUS = SPLB

PARAMETER C\_MEM3\_HIGHADDR = 0x00000000, DT = std\_logic\_vector, PAIR = C\_MEM3\_BASEADDR, ADDRESS = HIGH, BUS = SPLB

PARAMETER C\_MEM4\_BASEADDR = 0xffffffff, DT = std\_logic\_vector, PAIR = C\_MEM4\_HIGHADDR, ADDRESS = BASE, BUS = SPLB

PARAMETER C\_MEM4\_HIGHADDR = 0x00000000, DT = std\_logic\_vector, PAIR = C\_MEM4\_BASEADDR, ADDRESS = HIGH, BUS = SPLB

## Ports

PORT data\_from\_user0 = "", DIR = I, VEC = [0:15]

PORT data\_from\_user1 = "", DIR = I, VEC = [0:15]

PORT data\_from\_user2 = "", DIR = I, VEC = [0:15]

PORT data\_from\_user3 = "", DIR = I, VEC = [0:15]

PORT data\_from\_user4 = "", DIR = I, VEC = [0:15]

PORT data\_to\_user = "", DIR = O, VEC = [0:15]

PORT user\_cs = "", DIR = O, VEC = [0:4]

PORT user\_rd = "", DIR = O

PORT user\_wr = "", DIR = O

PORT user\_rst = "", DIR = O

PORT user\_clk = "", DIR = O

PORT user\_add = "", DIR = O, VEC = [0:11]

PORT SPLB\_Clk = "", DIR = I, SIGIS = CLK, BUS = SPLB

PORT SPLB\_Rst = SPLB\_Rst, DIR = I, SIGIS = RST, BUS = SPLB

PORT PLB\_ABus = PLB\_ABus, DIR = I, VEC = [0:31], BUS = SPLB

PORT PLB\_UABus = PLB\_UABus, DIR = I, VEC = [0:31], BUS = SPLB

PORT PLB\_PAValid = PLB\_PAValid, DIR = I, BUS = SPLB

PORT PLB\_SAValid = PLB\_SAValid, DIR = I, BUS = SPLB

PORT PLB\_rdPrim = PLB\_rdPrim, DIR = I, BUS = SPLB

PORT PLB\_wrPrim = PLB\_wrPrim, DIR = I, BUS = SPLB

PORT PLB\_masterID = PLB\_masterID, DIR = I, VEC = [0:(C\_SPLB\_MID\_WIDTH-1)], BUS = SPLB

PORT PLB\_abort = PLB\_abort, DIR = I, BUS = SPLB

PORT PLB\_busLock = PLB\_busLock, DIR = I, BUS = SPLB

PORT PLB\_RNW = PLB\_RNW, DIR = I, BUS = SPLB

PORT PLB\_BE = PLB\_BE, DIR = I, VEC = [0:((C\_SPLB\_DWIDTH/8)-1)], BUS = SPLB

PORT PLB\_MSize = PLB\_MSize, DIR = I, VEC = [0:1], BUS = SPLB

PORT PLB\_size = PLB\_size, DIR = I, VEC = [0:3], BUS = SPLB

PORT PLB\_type = PLB\_type, DIR = I, VEC = [0:2], BUS = SPLB

PORT PLB\_lockErr = PLB\_lockErr, DIR = I, BUS = SPLB

PORT PLB\_wrDBus = PLB\_wrDBus, DIR = I, VEC = [0:(C\_SPLB\_DWIDTH-1)], BUS = SPLB

PORT PLB\_wrBurst = PLB\_wrBurst, DIR = I, BUS = SPLB

PORT PLB\_rdBurst = PLB\_rdBurst, DIR = I, BUS = SPLB

PORT PLB\_wrPendReq = PLB\_wrPendReq, DIR = I, BUS = SPLB

PORT PLB\_rdPendReq = PLB\_rdPendReq, DIR = I, BUS = SPLB

PORT PLB\_wrPendPri = PLB\_wrPendPri, DIR = I, VEC = [0:1], BUS = SPLB

PORT PLB\_rdPendPri = PLB\_rdPendPri, DIR = I, VEC = [0:1], BUS = SPLB

PORT PLB\_reqPri = PLB\_reqPri, DIR = I, VEC = [0:1], BUS = SPLB

PORT PLB\_TAttribute = PLB\_TAttribute, DIR = I, VEC = [0:15], BUS = SPLB

PORT Sl\_addrAck = Sl\_addrAck, DIR = O, BUS = SPLB

PORT Sl\_SSize = Sl\_SSize, DIR = O, VEC = [0:1], BUS = SPLB

PORT Sl\_wait = Sl\_wait, DIR = O, BUS = SPLB

PORT Sl\_rearbitrate = Sl\_rearbitrate, DIR = O, BUS = SPLB

PORT Sl\_wrDAck = Sl\_wrDAck, DIR = O, BUS = SPLB

PORT Sl\_wrComp = Sl\_wrComp, DIR = O, BUS = SPLB

PORT Sl\_wrBTerm = Sl\_wrBTerm, DIR = O, BUS = SPLB

PORT Sl\_rdDBus = Sl\_rdDBus, DIR = O, VEC = [0:(C\_SPLB\_DWIDTH-1)], BUS = SPLB

PORT Sl\_rdWdAddr = Sl\_rdWdAddr, DIR = O, VEC = [0:3], BUS = SPLB

PORT Sl\_rdDAck = Sl\_rdDAck, DIR = O, BUS = SPLB

PORT Sl\_rdComp = Sl\_rdComp, DIR = O, BUS = SPLB

PORT Sl\_rdBTerm = Sl\_rdBTerm, DIR = O, BUS = SPLB

PORT Sl\_MBusy = Sl\_MBusy, DIR = O, VEC = [0:(C\_SPLB\_NUM\_MASTERS-1)], BUS = SPLB

PORT Sl\_MWrErr = Sl\_MWrErr, DIR = O, VEC = [0:(C\_SPLB\_NUM\_MASTERS-1)], BUS = SPLB

PORT Sl\_MRdErr = Sl\_MRdErr, DIR = O, VEC = [0:(C\_SPLB\_NUM\_MASTERS-1)], BUS = SPLB

PORT Sl\_MIRQ = Sl\_MIRQ, DIR = O, VEC = [0:(C\_SPLB\_NUM\_MASTERS-1)], BUS = SPLB

END

------------------------------------------------------------------------------

-- plb\_user\_logic\_16x8192\_4cs.vhd - entity/architecture pair

------------------------------------------------------------------------------

-- IMPORTANT:

-- DO NOT MODIFY THIS FILE EXCEPT IN THE DESIGNATED SECTIONS.

--

-- SEARCH FOR --USER TO DETERMINE WHERE CHANGES ARE ALLOWED.

--

-- TYPICALLY, THE ONLY ACCEPTABLE CHANGES INVOLVE ADDING NEW

-- PORTS AND GENERICS THAT GET PASSED THROUGH TO THE INSTANTIATION

-- OF THE USER\_LOGIC ENTITY.

------------------------------------------------------------------------------

--

-- \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

-- \*\* Copyright (c) 1995-2008 Xilinx, Inc. All rights reserved. \*\*

-- \*\* \*\*

-- \*\* Xilinx, Inc. \*\*

-- \*\* XILINX IS PROVIDING THIS DESIGN, CODE, OR INFORMATION "AS IS" \*\*

-- \*\* AS A COURTESY TO YOU, SOLELY FOR USE IN DEVELOPING PROGRAMS AND \*\*

-- \*\* SOLUTIONS FOR XILINX DEVICES. BY PROVIDING THIS DESIGN, CODE, \*\*

-- \*\* OR INFORMATION AS ONE POSSIBLE IMPLEMENTATION OF THIS FEATURE, \*\*

-- \*\* APPLICATION OR STANDARD, XILINX IS MAKING NO REPRESENTATION \*\*

-- \*\* THAT THIS IMPLEMENTATION IS FREE FROM ANY CLAIMS OF INFRINGEMENT, \*\*

-- \*\* AND YOU ARE RESPONSIBLE FOR OBTAINING ANY RIGHTS YOU MAY REQUIRE \*\*

-- \*\* FOR YOUR IMPLEMENTATION. XILINX EXPRESSLY DISCLAIMS ANY \*\*

-- \*\* WARRANTY WHATSOEVER WITH RESPECT TO THE ADEQUACY OF THE \*\*

-- \*\* IMPLEMENTATION, INCLUDING BUT NOT LIMITED TO ANY WARRANTIES OR \*\*

-- \*\* REPRESENTATIONS THAT THIS IMPLEMENTATION IS FREE FROM CLAIMS OF \*\*

-- \*\* INFRINGEMENT, IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS \*\*

-- \*\* FOR A PARTICULAR PURPOSE. \*\*

-- \*\* \*\*

-- \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

--

------------------------------------------------------------------------------

-- Filename: plb\_user\_logic\_16x8192\_4cs.vhd

-- Version: 1.00.a

-- Description: Top level design, instantiates library components and user logic.

-- Date: Tue Jun 09 08:53:19 2009 (by Create and Import Peripheral Wizard)

-- VHDL Standard: VHDL'93

------------------------------------------------------------------------------

-- Naming Conventions:

-- active low signals: "\*\_n"

-- clock signals: "clk", "clk\_div#", "clk\_#x"

-- reset signals: "rst", "rst\_n"

-- generics: "C\_\*"

-- user defined types: "\*\_TYPE"

-- state machine next state: "\*\_ns"

-- state machine current state: "\*\_cs"

-- combinatorial signals: "\*\_com"

-- pipelined or register delay signals: "\*\_d#"

-- counter signals: "\*cnt\*"

-- clock enable signals: "\*\_ce"

-- internal version of output port: "\*\_i"

-- device pins: "\*\_pin"

-- ports: "- Names begin with Uppercase"

-- processes: "\*\_PROCESS"

-- component instantiations: "<ENTITY\_>I\_<#|FUNC>"

------------------------------------------------------------------------------

library ieee;

use ieee.std\_logic\_1164.all;

use ieee.std\_logic\_arith.all;

use ieee.std\_logic\_unsigned.all;

library proc\_common\_v2\_00\_a;

use proc\_common\_v2\_00\_a.proc\_common\_pkg.all;

use proc\_common\_v2\_00\_a.ipif\_pkg.all;

library plbv46\_slave\_single\_v1\_00\_a;

use plbv46\_slave\_single\_v1\_00\_a.plbv46\_slave\_single;

library plb\_user\_logic\_16x8192\_4cs\_v1\_00\_a;

use plb\_user\_logic\_16x8192\_4cs\_v1\_00\_a.user\_logic;

------------------------------------------------------------------------------

-- Entity section

------------------------------------------------------------------------------

-- Definition of Generics:

-- C\_SPLB\_AWIDTH -- PLBv46 slave: address bus width

-- C\_SPLB\_DWIDTH -- PLBv46 slave: data bus width

-- C\_SPLB\_NUM\_MASTERS -- PLBv46 slave: Number of masters

-- C\_SPLB\_MID\_WIDTH -- PLBv46 slave: master ID bus width

-- C\_SPLB\_NATIVE\_DWIDTH -- PLBv46 slave: internal native data bus width

-- C\_SPLB\_P2P -- PLBv46 slave: point to point interconnect scheme

-- C\_SPLB\_SUPPORT\_BURSTS -- PLBv46 slave: support bursts

-- C\_SPLB\_SMALLEST\_MASTER -- PLBv46 slave: width of the smallest master

-- C\_SPLB\_CLK\_PERIOD\_PS -- PLBv46 slave: bus clock in picoseconds

-- C\_INCLUDE\_DPHASE\_TIMER -- PLBv46 slave: Data Phase Timer configuration; 0 = exclude timer, 1 = include timer

-- C\_FAMILY -- Xilinx FPGA family

-- C\_MEM0\_BASEADDR -- User memory space 0 base address

-- C\_MEM0\_HIGHADDR -- User memory space 0 high address

-- C\_MEM1\_BASEADDR -- User memory space 1 base address

-- C\_MEM1\_HIGHADDR -- User memory space 1 high address

-- C\_MEM2\_BASEADDR -- User memory space 2 base address

-- C\_MEM2\_HIGHADDR -- User memory space 2 high address

-- C\_MEM3\_BASEADDR -- User memory space 3 base address

-- C\_MEM3\_HIGHADDR -- User memory space 3 high address

--

-- Definition of Ports:

-- SPLB\_Clk -- PLB main bus clock

-- SPLB\_Rst -- PLB main bus reset

-- PLB\_ABus -- PLB address bus

-- PLB\_UABus -- PLB upper address bus

-- PLB\_PAValid -- PLB primary address valid indicator

-- PLB\_SAValid -- PLB secondary address valid indicator

-- PLB\_rdPrim -- PLB secondary to primary read request indicator

-- PLB\_wrPrim -- PLB secondary to primary write request indicator

-- PLB\_masterID -- PLB current master identifier

-- PLB\_abort -- PLB abort request indicator

-- PLB\_busLock -- PLB bus lock

-- PLB\_RNW -- PLB read/not write

-- PLB\_BE -- PLB byte enables

-- PLB\_MSize -- PLB master data bus size

-- PLB\_size -- PLB transfer size

-- PLB\_type -- PLB transfer type

-- PLB\_lockErr -- PLB lock error indicator

-- PLB\_wrDBus -- PLB write data bus

-- PLB\_wrBurst -- PLB burst write transfer indicator

-- PLB\_rdBurst -- PLB burst read transfer indicator

-- PLB\_wrPendReq -- PLB write pending bus request indicator

-- PLB\_rdPendReq -- PLB read pending bus request indicator

-- PLB\_wrPendPri -- PLB write pending request priority

-- PLB\_rdPendPri -- PLB read pending request priority

-- PLB\_reqPri -- PLB current request priority

-- PLB\_TAttribute -- PLB transfer attribute

-- Sl\_addrAck -- Slave address acknowledge

-- Sl\_SSize -- Slave data bus size

-- Sl\_wait -- Slave wait indicator

-- Sl\_rearbitrate -- Slave re-arbitrate bus indicator

-- Sl\_wrDAck -- Slave write data acknowledge

-- Sl\_wrComp -- Slave write transfer complete indicator

-- Sl\_wrBTerm -- Slave terminate write burst transfer

-- Sl\_rdDBus -- Slave read data bus

-- Sl\_rdWdAddr -- Slave read word address

-- Sl\_rdDAck -- Slave read data acknowledge

-- Sl\_rdComp -- Slave read transfer complete indicator

-- Sl\_rdBTerm -- Slave terminate read burst transfer

-- Sl\_MBusy -- Slave busy indicator

-- Sl\_MWrErr -- Slave write error indicator

-- Sl\_MRdErr -- Slave read error indicator

-- Sl\_MIRQ -- Slave interrupt indicator

------------------------------------------------------------------------------

entity plb\_user\_logic\_16x8192\_4cs is

generic

(

-- ADD USER GENERICS BELOW THIS LINE ---------------

--USER generics added here

-- ADD USER GENERICS ABOVE THIS LINE ---------------

-- DO NOT EDIT BELOW THIS LINE ---------------------

-- Bus protocol parameters, do not add to or delete

C\_SPLB\_AWIDTH : integer := 32;

C\_SPLB\_DWIDTH : integer := 128;

C\_SPLB\_NUM\_MASTERS : integer := 8;

C\_SPLB\_MID\_WIDTH : integer := 3;

C\_SPLB\_NATIVE\_DWIDTH : integer := 32;

C\_SPLB\_P2P : integer := 0;

C\_SPLB\_SUPPORT\_BURSTS : integer := 0;

C\_SPLB\_SMALLEST\_MASTER : integer := 32;

C\_SPLB\_CLK\_PERIOD\_PS : integer := 10000;

C\_INCLUDE\_DPHASE\_TIMER : integer := 0;

C\_FAMILY : string := "virtex5";

C\_MEM0\_BASEADDR : std\_logic\_vector := X"FFFFFFFF";

C\_MEM0\_HIGHADDR : std\_logic\_vector := X"00000000";

C\_MEM1\_BASEADDR : std\_logic\_vector := X"FFFFFFFF";

C\_MEM1\_HIGHADDR : std\_logic\_vector := X"00000000";

C\_MEM2\_BASEADDR : std\_logic\_vector := X"FFFFFFFF";

C\_MEM2\_HIGHADDR : std\_logic\_vector := X"00000000";

C\_MEM3\_BASEADDR : std\_logic\_vector := X"FFFFFFFF";

C\_MEM3\_HIGHADDR : std\_logic\_vector := X"00000000"

-- DO NOT EDIT ABOVE THIS LINE ---------------------

);

port

(

-- ADD USER PORTS BELOW THIS LINE ------------------

--USER ports added here

data\_from\_user0 : in std\_logic\_vector(0 to 15);

data\_from\_user1 : in std\_logic\_vector(0 to 15);

data\_from\_user2 : in std\_logic\_vector(0 to 15);

data\_from\_user3 : in std\_logic\_vector(0 to 15);

data\_to\_user : out std\_logic\_vector(0 to 15);

user\_cs : out std\_logic\_vector(0 to 3);

user\_rd : out std\_logic;

user\_wr : out std\_logic;

user\_rst : out std\_logic;

user\_clk : out std\_logic;

user\_add : out std\_logic\_vector(0 to 11);

-- ADD USER PORTS ABOVE THIS LINE ------------------

-- DO NOT EDIT BELOW THIS LINE ---------------------

-- Bus protocol ports, do not add to or delete

SPLB\_Clk : in std\_logic;

SPLB\_Rst : in std\_logic;

PLB\_ABus : in std\_logic\_vector(0 to 31);

PLB\_UABus : in std\_logic\_vector(0 to 31);

PLB\_PAValid : in std\_logic;

PLB\_SAValid : in std\_logic;

PLB\_rdPrim : in std\_logic;

PLB\_wrPrim : in std\_logic;

PLB\_masterID : in std\_logic\_vector(0 to C\_SPLB\_MID\_WIDTH-1);

PLB\_abort : in std\_logic;

PLB\_busLock : in std\_logic;

PLB\_RNW : in std\_logic;

PLB\_BE : in std\_logic\_vector(0 to C\_SPLB\_DWIDTH/8-1);

PLB\_MSize : in std\_logic\_vector(0 to 1);

PLB\_size : in std\_logic\_vector(0 to 3);

PLB\_type : in std\_logic\_vector(0 to 2);

PLB\_lockErr : in std\_logic;

PLB\_wrDBus : in std\_logic\_vector(0 to C\_SPLB\_DWIDTH-1);

PLB\_wrBurst : in std\_logic;

PLB\_rdBurst : in std\_logic;

PLB\_wrPendReq : in std\_logic;

PLB\_rdPendReq : in std\_logic;

PLB\_wrPendPri : in std\_logic\_vector(0 to 1);

PLB\_rdPendPri : in std\_logic\_vector(0 to 1);

PLB\_reqPri : in std\_logic\_vector(0 to 1);

PLB\_TAttribute : in std\_logic\_vector(0 to 15);

Sl\_addrAck : out std\_logic;

Sl\_SSize : out std\_logic\_vector(0 to 1);

Sl\_wait : out std\_logic;

Sl\_rearbitrate : out std\_logic;

Sl\_wrDAck : out std\_logic;

Sl\_wrComp : out std\_logic;

Sl\_wrBTerm : out std\_logic;

Sl\_rdDBus : out std\_logic\_vector(0 to C\_SPLB\_DWIDTH-1);

Sl\_rdWdAddr : out std\_logic\_vector(0 to 3);

Sl\_rdDAck : out std\_logic;

Sl\_rdComp : out std\_logic;

Sl\_rdBTerm : out std\_logic;

Sl\_MBusy : out std\_logic\_vector(0 to C\_SPLB\_NUM\_MASTERS-1);

Sl\_MWrErr : out std\_logic\_vector(0 to C\_SPLB\_NUM\_MASTERS-1);

Sl\_MRdErr : out std\_logic\_vector(0 to C\_SPLB\_NUM\_MASTERS-1);

Sl\_MIRQ : out std\_logic\_vector(0 to C\_SPLB\_NUM\_MASTERS-1)

-- DO NOT EDIT ABOVE THIS LINE ---------------------

);

attribute SIGIS : string;

attribute SIGIS of SPLB\_Clk : signal is "CLK";

attribute SIGIS of SPLB\_Rst : signal is "RST";

end entity plb\_user\_logic\_16x8192\_4cs;

------------------------------------------------------------------------------

-- Architecture section

------------------------------------------------------------------------------

architecture IMP of plb\_user\_logic\_16x8192\_4cs is

------------------------------------------

-- Array of base/high address pairs for each address range

------------------------------------------

constant ZERO\_ADDR\_PAD : std\_logic\_vector(0 to 31) := (others => '0');

constant IPIF\_ARD\_ADDR\_RANGE\_ARRAY : SLV64\_ARRAY\_TYPE :=

(

ZERO\_ADDR\_PAD & C\_MEM0\_BASEADDR, -- user logic memory space 0 base address

ZERO\_ADDR\_PAD & C\_MEM0\_HIGHADDR, -- user logic memory space 0 high address

ZERO\_ADDR\_PAD & C\_MEM1\_BASEADDR, -- user logic memory space 1 base address

ZERO\_ADDR\_PAD & C\_MEM1\_HIGHADDR, -- user logic memory space 1 high address

ZERO\_ADDR\_PAD & C\_MEM2\_BASEADDR, -- user logic memory space 2 base address

ZERO\_ADDR\_PAD & C\_MEM2\_HIGHADDR, -- user logic memory space 2 high address

ZERO\_ADDR\_PAD & C\_MEM3\_BASEADDR, -- user logic memory space 3 base address

ZERO\_ADDR\_PAD & C\_MEM3\_HIGHADDR -- user logic memory space 3 high address

);

------------------------------------------

-- Array of desired number of chip enables for each address range

------------------------------------------

constant USER\_NUM\_MEM : integer := 4;

constant IPIF\_ARD\_NUM\_CE\_ARRAY : INTEGER\_ARRAY\_TYPE :=

(

0 => 1, -- number of ce for user logic memory space 0 (always 1 chip enable)

1 => 1, -- number of ce for user logic memory space 1 (always 1 chip enable)

2 => 1, -- number of ce for user logic memory space 2 (always 1 chip enable)

3 => 1 -- number of ce for user logic memory space 3 (always 1 chip enable)

);

------------------------------------------

-- Ratio of bus clock to core clock (for use in dual clock systems)

-- 1 = ratio is 1:1

-- 2 = ratio is 2:1

------------------------------------------

constant IPIF\_BUS2CORE\_CLK\_RATIO : integer := 1;

------------------------------------------

-- Width of the slave data bus (32 only)

------------------------------------------

constant USER\_SLV\_DWIDTH : integer := C\_SPLB\_NATIVE\_DWIDTH;

constant IPIF\_SLV\_DWIDTH : integer := C\_SPLB\_NATIVE\_DWIDTH;

------------------------------------------

-- Width of the slave address bus (32 only)

------------------------------------------

constant USER\_SLV\_AWIDTH : integer := C\_SPLB\_AWIDTH;

------------------------------------------

-- Index for CS/CE

------------------------------------------

constant USER\_MEM0\_CS\_INDEX : integer := 0;

constant USER\_CS\_INDEX : integer := USER\_MEM0\_CS\_INDEX;

------------------------------------------

-- IP Interconnect (IPIC) signal declarations

------------------------------------------

signal ipif\_Bus2IP\_Clk : std\_logic;

signal ipif\_Bus2IP\_Reset : std\_logic;

signal ipif\_IP2Bus\_Data : std\_logic\_vector(0 to IPIF\_SLV\_DWIDTH-1);

signal ipif\_IP2Bus\_WrAck : std\_logic;

signal ipif\_IP2Bus\_RdAck : std\_logic;

signal ipif\_IP2Bus\_Error : std\_logic;

signal ipif\_Bus2IP\_Addr : std\_logic\_vector(0 to C\_SPLB\_AWIDTH-1);

signal ipif\_Bus2IP\_Data : std\_logic\_vector(0 to IPIF\_SLV\_DWIDTH-1);

signal ipif\_Bus2IP\_RNW : std\_logic;

signal ipif\_Bus2IP\_BE : std\_logic\_vector(0 to IPIF\_SLV\_DWIDTH/8-1);

signal ipif\_Bus2IP\_CS : std\_logic\_vector(0 to ((IPIF\_ARD\_ADDR\_RANGE\_ARRAY'length)/2)-1);

signal ipif\_Bus2IP\_RdCE : std\_logic\_vector(0 to calc\_num\_ce(IPIF\_ARD\_NUM\_CE\_ARRAY)-1);

signal ipif\_Bus2IP\_WrCE : std\_logic\_vector(0 to calc\_num\_ce(IPIF\_ARD\_NUM\_CE\_ARRAY)-1);

signal user\_IP2Bus\_Data : std\_logic\_vector(0 to USER\_SLV\_DWIDTH-1);

signal user\_IP2Bus\_RdAck : std\_logic;

signal user\_IP2Bus\_WrAck : std\_logic;

signal user\_IP2Bus\_Error : std\_logic;

begin

------------------------------------------

-- instantiate plbv46\_slave\_single

------------------------------------------

PLBV46\_SLAVE\_SINGLE\_I : entity plbv46\_slave\_single\_v1\_00\_a.plbv46\_slave\_single

generic map

(

C\_ARD\_ADDR\_RANGE\_ARRAY => IPIF\_ARD\_ADDR\_RANGE\_ARRAY,

C\_ARD\_NUM\_CE\_ARRAY => IPIF\_ARD\_NUM\_CE\_ARRAY,

C\_SPLB\_P2P => C\_SPLB\_P2P,

C\_BUS2CORE\_CLK\_RATIO => IPIF\_BUS2CORE\_CLK\_RATIO,

C\_SPLB\_MID\_WIDTH => C\_SPLB\_MID\_WIDTH,

C\_SPLB\_NUM\_MASTERS => C\_SPLB\_NUM\_MASTERS,

C\_SPLB\_AWIDTH => C\_SPLB\_AWIDTH,

C\_SPLB\_DWIDTH => C\_SPLB\_DWIDTH,

C\_SIPIF\_DWIDTH => IPIF\_SLV\_DWIDTH,

C\_INCLUDE\_DPHASE\_TIMER => C\_INCLUDE\_DPHASE\_TIMER,

C\_FAMILY => C\_FAMILY

)

port map

(

SPLB\_Clk => SPLB\_Clk,

SPLB\_Rst => SPLB\_Rst,

PLB\_ABus => PLB\_ABus,

PLB\_UABus => PLB\_UABus,

PLB\_PAValid => PLB\_PAValid,

PLB\_SAValid => PLB\_SAValid,

PLB\_rdPrim => PLB\_rdPrim,

PLB\_wrPrim => PLB\_wrPrim,

PLB\_masterID => PLB\_masterID,

PLB\_abort => PLB\_abort,

PLB\_busLock => PLB\_busLock,

PLB\_RNW => PLB\_RNW,

PLB\_BE => PLB\_BE,

PLB\_MSize => PLB\_MSize,

PLB\_size => PLB\_size,

PLB\_type => PLB\_type,

PLB\_lockErr => PLB\_lockErr,

PLB\_wrDBus => PLB\_wrDBus,

PLB\_wrBurst => PLB\_wrBurst,

PLB\_rdBurst => PLB\_rdBurst,

PLB\_wrPendReq => PLB\_wrPendReq,

PLB\_rdPendReq => PLB\_rdPendReq,

PLB\_wrPendPri => PLB\_wrPendPri,

PLB\_rdPendPri => PLB\_rdPendPri,

PLB\_reqPri => PLB\_reqPri,

PLB\_TAttribute => PLB\_TAttribute,

Sl\_addrAck => Sl\_addrAck,

Sl\_SSize => Sl\_SSize,

Sl\_wait => Sl\_wait,

Sl\_rearbitrate => Sl\_rearbitrate,

Sl\_wrDAck => Sl\_wrDAck,

Sl\_wrComp => Sl\_wrComp,

Sl\_wrBTerm => Sl\_wrBTerm,

Sl\_rdDBus => Sl\_rdDBus,

Sl\_rdWdAddr => Sl\_rdWdAddr,

Sl\_rdDAck => Sl\_rdDAck,

Sl\_rdComp => Sl\_rdComp,

Sl\_rdBTerm => Sl\_rdBTerm,

Sl\_MBusy => Sl\_MBusy,

Sl\_MWrErr => Sl\_MWrErr,

Sl\_MRdErr => Sl\_MRdErr,

Sl\_MIRQ => Sl\_MIRQ,

Bus2IP\_Clk => ipif\_Bus2IP\_Clk,

Bus2IP\_Reset => ipif\_Bus2IP\_Reset,

IP2Bus\_Data => ipif\_IP2Bus\_Data,

IP2Bus\_WrAck => ipif\_IP2Bus\_WrAck,

IP2Bus\_RdAck => ipif\_IP2Bus\_RdAck,

IP2Bus\_Error => ipif\_IP2Bus\_Error,

Bus2IP\_Addr => ipif\_Bus2IP\_Addr,

Bus2IP\_Data => ipif\_Bus2IP\_Data,

Bus2IP\_RNW => ipif\_Bus2IP\_RNW,

Bus2IP\_BE => ipif\_Bus2IP\_BE,

Bus2IP\_CS => ipif\_Bus2IP\_CS,

Bus2IP\_RdCE => ipif\_Bus2IP\_RdCE,

Bus2IP\_WrCE => ipif\_Bus2IP\_WrCE

);

------------------------------------------

-- instantiate User Logic

------------------------------------------

USER\_LOGIC\_I : entity plb\_user\_logic\_16x8192\_4cs\_v1\_00\_a.user\_logic

generic map

(

-- MAP USER GENERICS BELOW THIS LINE ---------------

--USER generics mapped here

-- MAP USER GENERICS ABOVE THIS LINE ---------------

C\_SLV\_AWIDTH => USER\_SLV\_AWIDTH,

C\_SLV\_DWIDTH => USER\_SLV\_DWIDTH,

C\_NUM\_MEM => USER\_NUM\_MEM

)

port map

(

-- MAP USER PORTS BELOW THIS LINE ------------------

--USER ports mapped here

data\_from\_user0 => data\_from\_user0,

data\_from\_user1 => data\_from\_user1,

data\_from\_user2 => data\_from\_user2,

data\_from\_user3 => data\_from\_user3,

data\_to\_user => data\_to\_user,

user\_cs => user\_cs,

user\_rd => user\_rd,

user\_wr => user\_wr,

user\_rst => user\_rst,

user\_clk => user\_clk,

user\_add => user\_add,

-- MAP USER PORTS ABOVE THIS LINE ------------------

Bus2IP\_Clk => ipif\_Bus2IP\_Clk,

Bus2IP\_Reset => ipif\_Bus2IP\_Reset,

Bus2IP\_Addr => ipif\_Bus2IP\_Addr,

Bus2IP\_CS => ipif\_Bus2IP\_CS(USER\_CS\_INDEX to USER\_CS\_INDEX+USER\_NUM\_MEM-1),

Bus2IP\_RNW => ipif\_Bus2IP\_RNW,

Bus2IP\_Data => ipif\_Bus2IP\_Data,

Bus2IP\_BE => ipif\_Bus2IP\_BE,

IP2Bus\_Data => user\_IP2Bus\_Data,

IP2Bus\_RdAck => user\_IP2Bus\_RdAck,

IP2Bus\_WrAck => user\_IP2Bus\_WrAck,

IP2Bus\_Error => user\_IP2Bus\_Error

);

------------------------------------------

-- connect internal signals

------------------------------------------

IP2BUS\_DATA\_MUX\_PROC : process( ipif\_Bus2IP\_CS, user\_IP2Bus\_Data ) is

begin

case ipif\_Bus2IP\_CS is

when "1000" => ipif\_IP2Bus\_Data <= user\_IP2Bus\_Data;

when "0100" => ipif\_IP2Bus\_Data <= user\_IP2Bus\_Data;

when "0010" => ipif\_IP2Bus\_Data <= user\_IP2Bus\_Data;

when "0001" => ipif\_IP2Bus\_Data <= user\_IP2Bus\_Data;

when others => ipif\_IP2Bus\_Data <= (others => '0');

end case;

end process IP2BUS\_DATA\_MUX\_PROC;

ipif\_IP2Bus\_WrAck <= user\_IP2Bus\_WrAck;

ipif\_IP2Bus\_RdAck <= user\_IP2Bus\_RdAck;

ipif\_IP2Bus\_Error <= user\_IP2Bus\_Error;

end IMP;







































