Adición de un periférico a un SoC

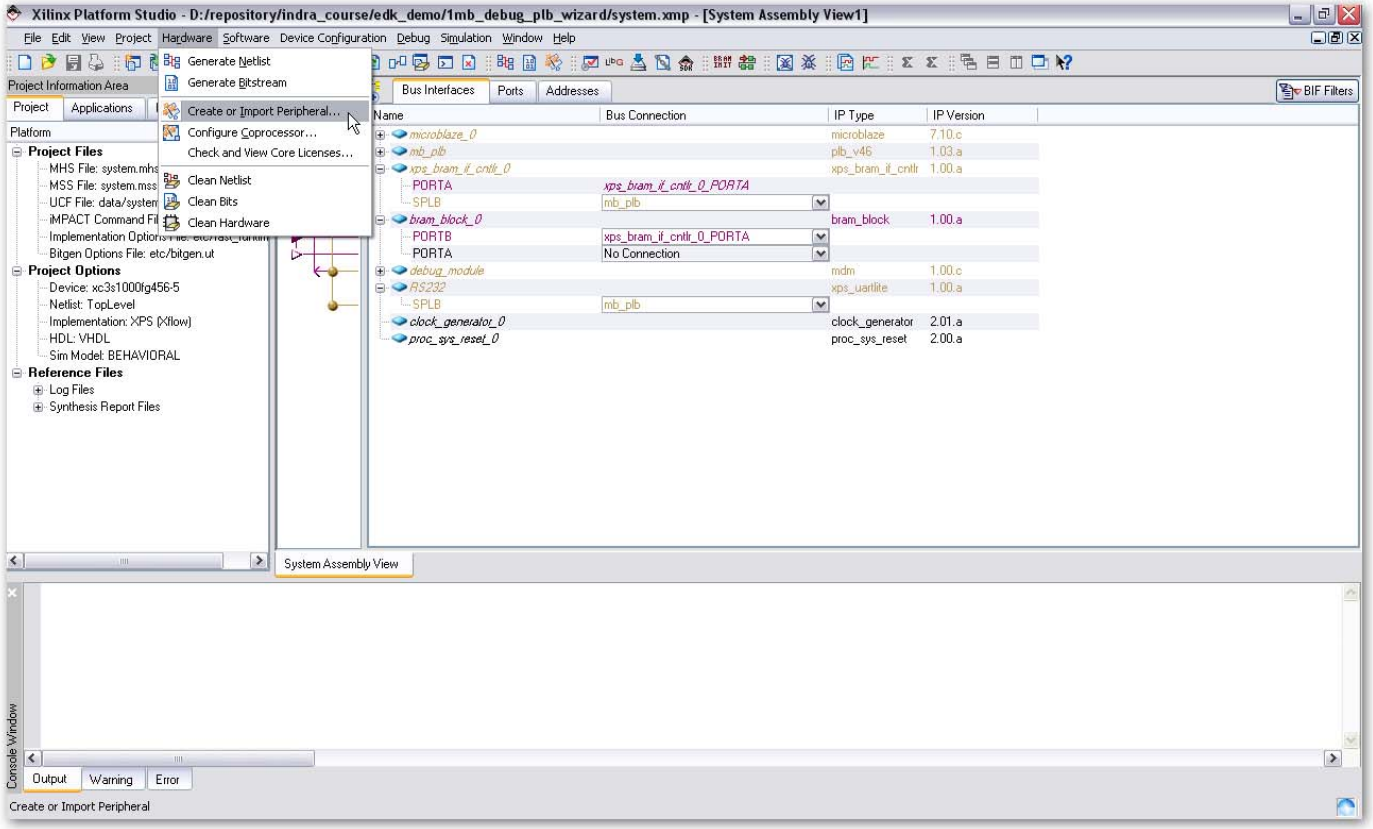
Hortensia Mecha

Pablo García del Valle, David Atienza Alonso

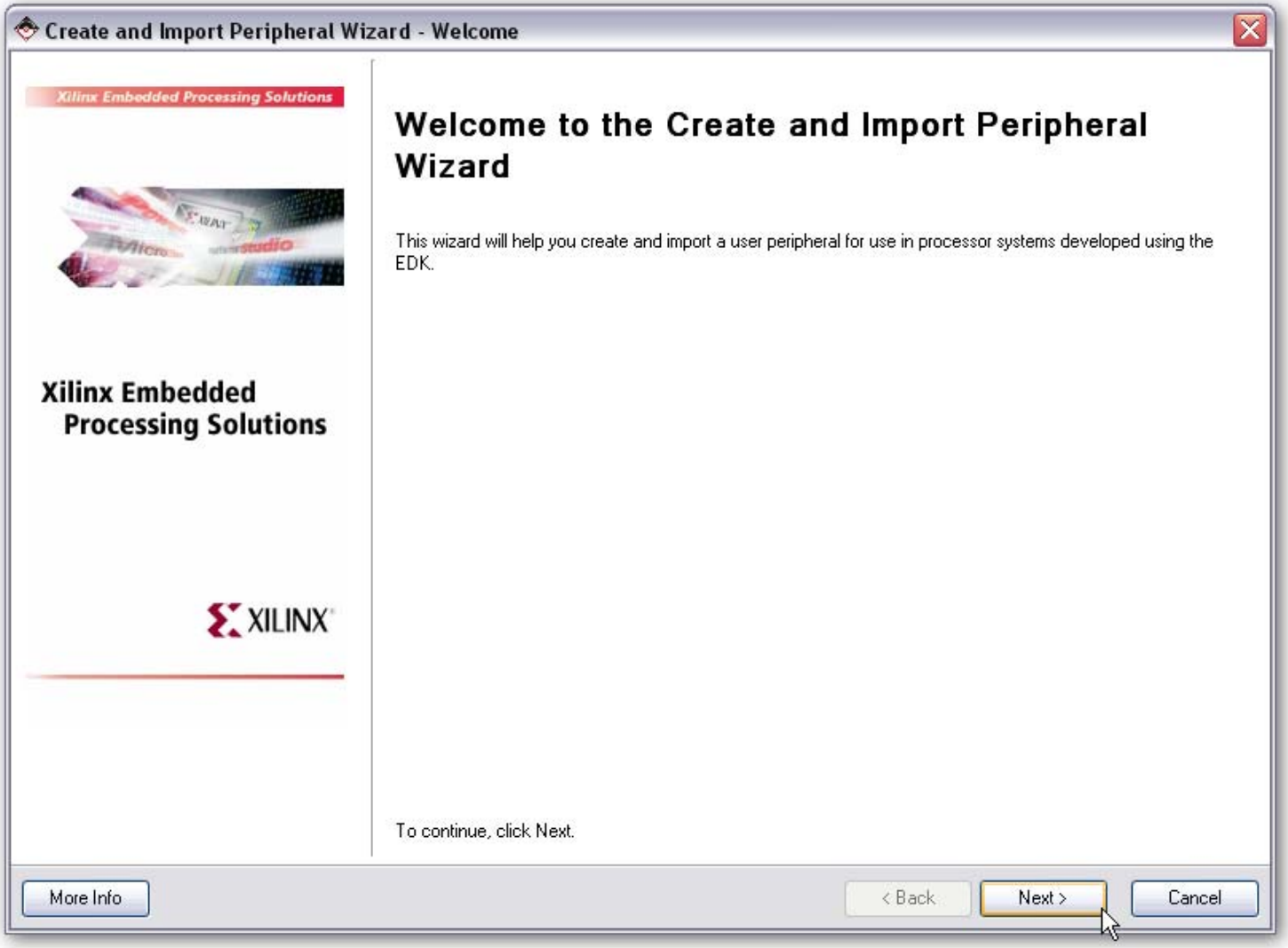
- Starting point – ISE project :

Utilización de la aplicación Create Peripheral Wizard

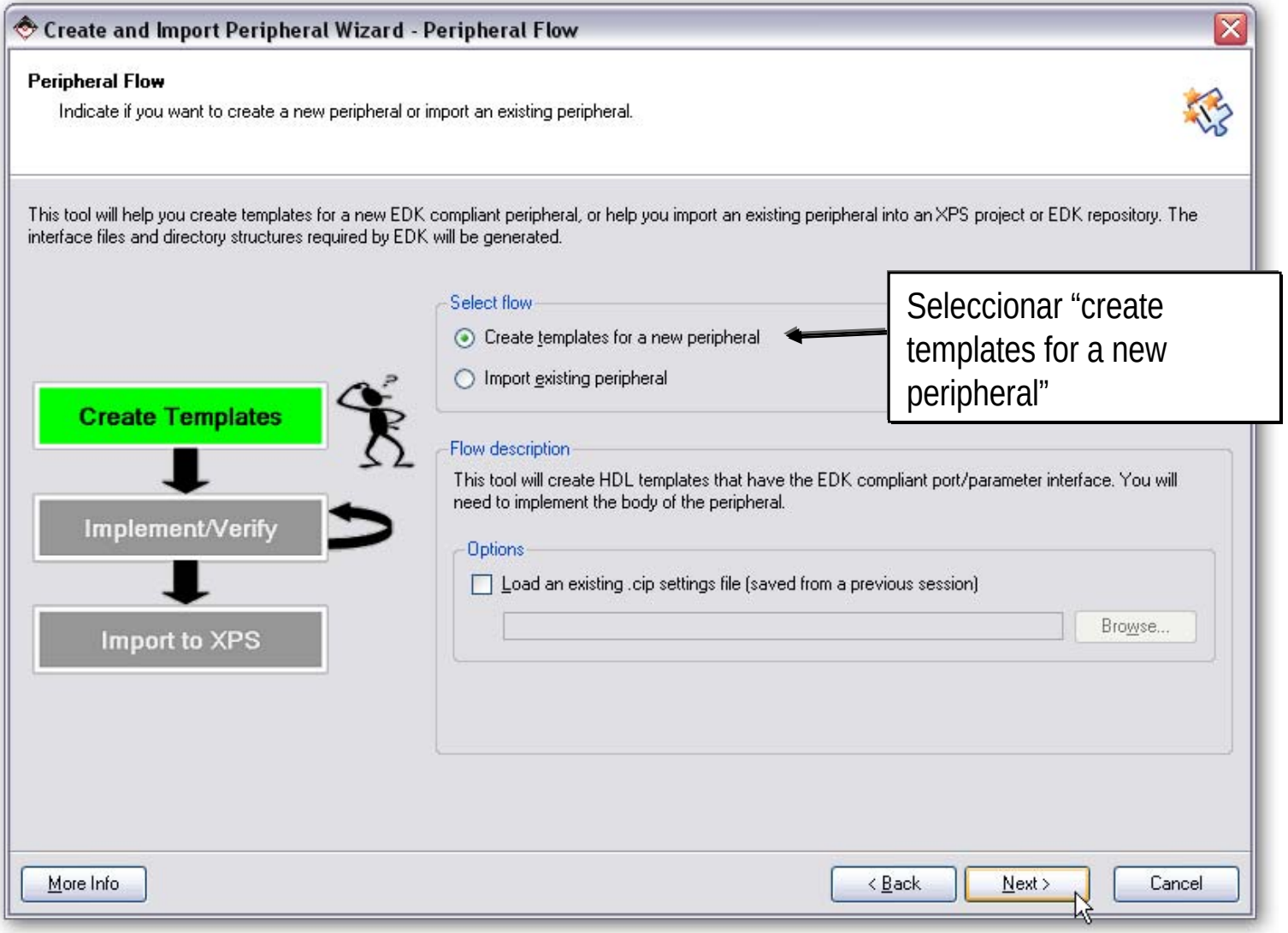
- Puede llamarse desde el menú programs en windows: EDK -> Accesories -> Create and Import Peripheral Wizard
- O desde dentro de EDK. Menu Hardware -> Create or Import Peripheral



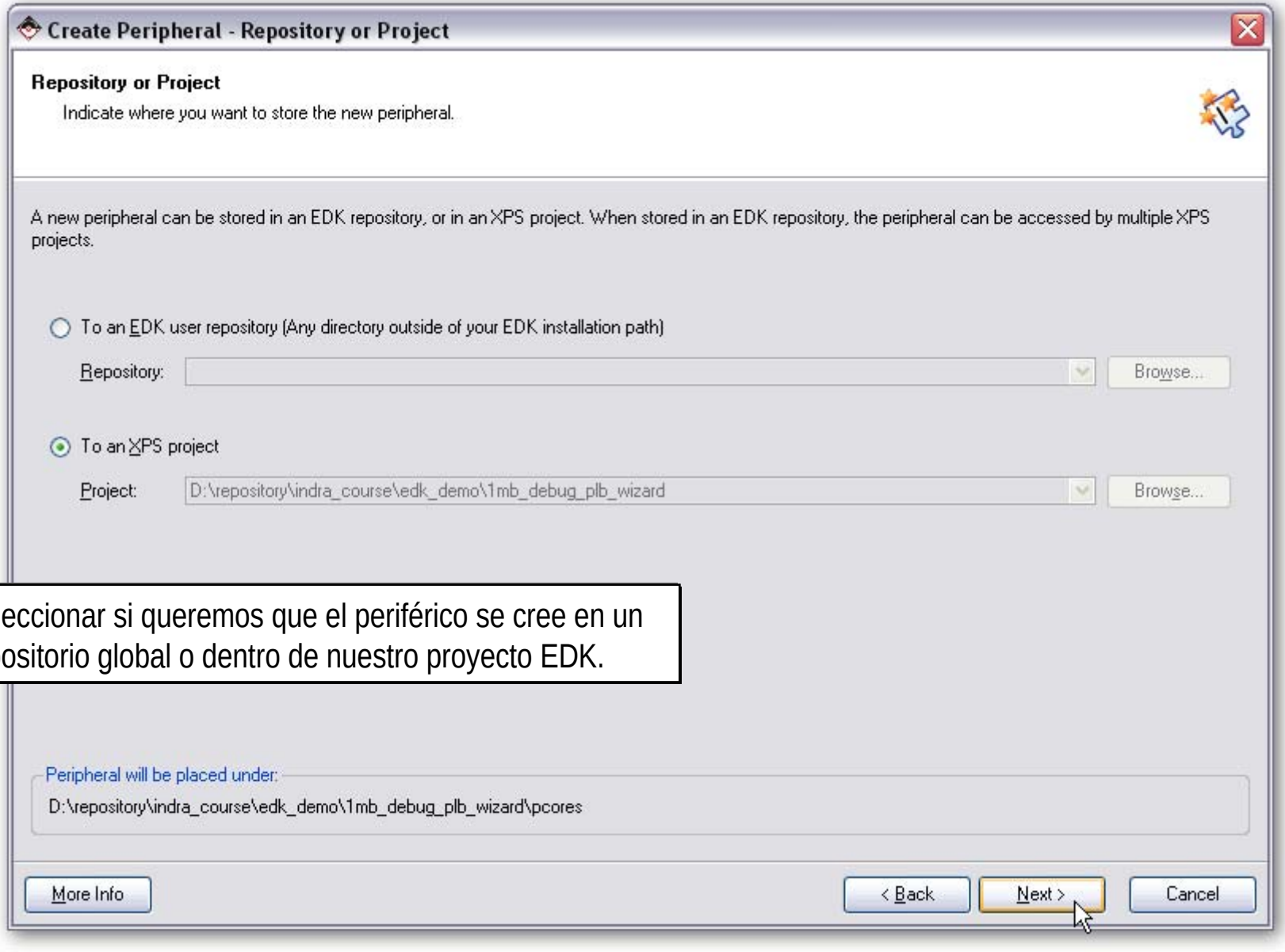
Utilización de la aplicación Create Peripheral Wizard



Utilización de la aplicación Create Peripheral Wizard

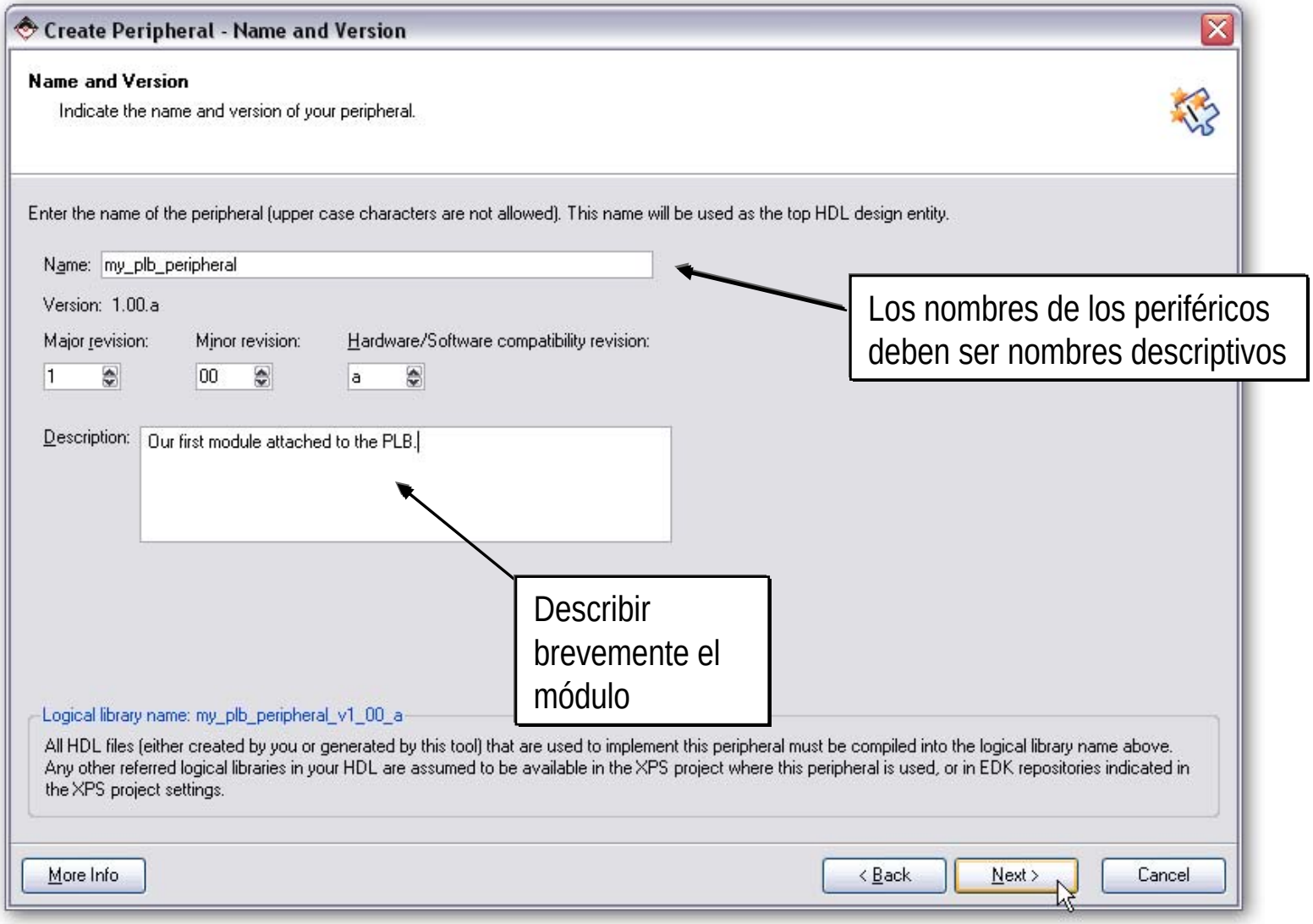


Utilización de la aplicación Create Peripheral Wizard



Seleccionar si queremos que el periférico se cree en un repositorio global o dentro de nuestro proyecto EDK.

Utilización de la aplicación Create Peripheral Wizard



Los nombres de los periféricos deben ser nombres descriptivos

Describir brevemente el módulo

Utilización de la aplicación Create Peripheral Wizard

Create Peripheral - Bus Interface

Bus Interface

Indicate the bus interface supported by your peripheral.

To which bus will this peripheral be attached?

- (●) Processor Local Bus (PLB v4.6)
- ( ) Fast Simplex Link (FSL)

ATTENTION

Refer to the following documents to get a better understanding of how user peripherals connect to the CoreConnect(TM) buses (including PLB v4.6 interconnect and OPB/PLB v3.4 interconnect) and the FSL interface.

NOTE - Select the bus interface above and the corresponding link(s) will appear below for that interface.

CoreConnect Specification

PLB (v4.6) Slave IPIF Specification for single data beat transfer

PLB (v4.6) Slave IPIF Specification for burst data transfer

PLB (v4.6) Master IPIF Specification for single data beat transfer

PLB (v4.6) Master IPIF Specification for burst data transfer

Note

Xilinx recommends using the new PLB v4.6 bus standard, however, the wizard still supports the OPB and PLB v3.4 bus interfaces.

[ ] Enable OPB and PLB v3.4 bus interfaces

More Info | < Back | Next > | Cancel

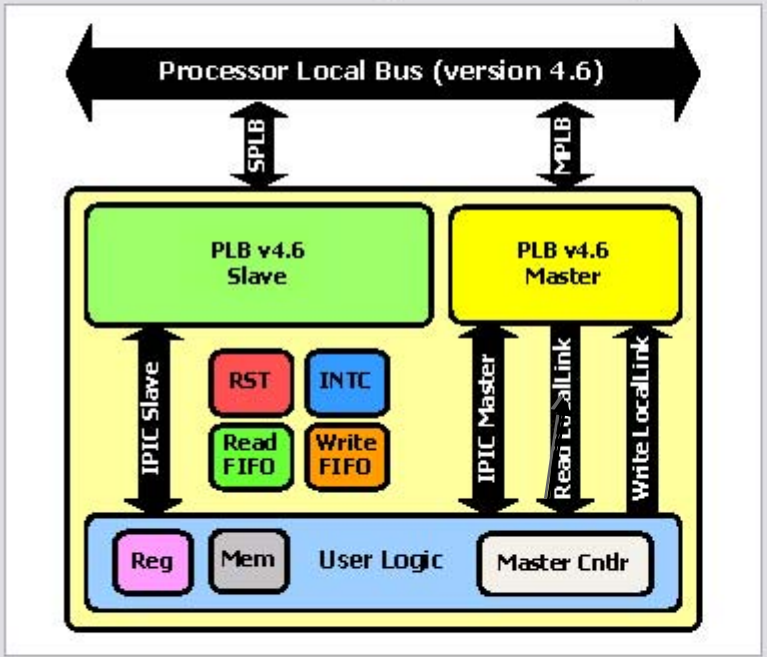
Utilización de la aplicación Create Peripheral Wizard

Create Peripheral - IPIF (IP Interface) Services

IPIF (IP Interface) Services

Indicate the IPIF services required by your peripheral.

Your peripheral will be connected to the PLB (v4.6) interconnect through corresponding PLB IP Interface (IPIF) modules, which provide you with a quick way to implement the interface between the PLB interconnect and the user logic. Besides the standard functions like address decoding provided by the slave IPIF module, the wizard tool also offers other commonly used services and configurations to simplify the implementation of the design.



Slave service and configuration

Typically required by most peripherals for operations like logic control, status report, data buffering, multiple memory/address space access, and etc. (PLB slave interface will always be included).

- [ ] Software reset
- [x] User logic software register
- [ ] Read/Write FIFO
- [ ] User logic memory space
- [ ] Interrupt control
- [ ] Include data phase timer

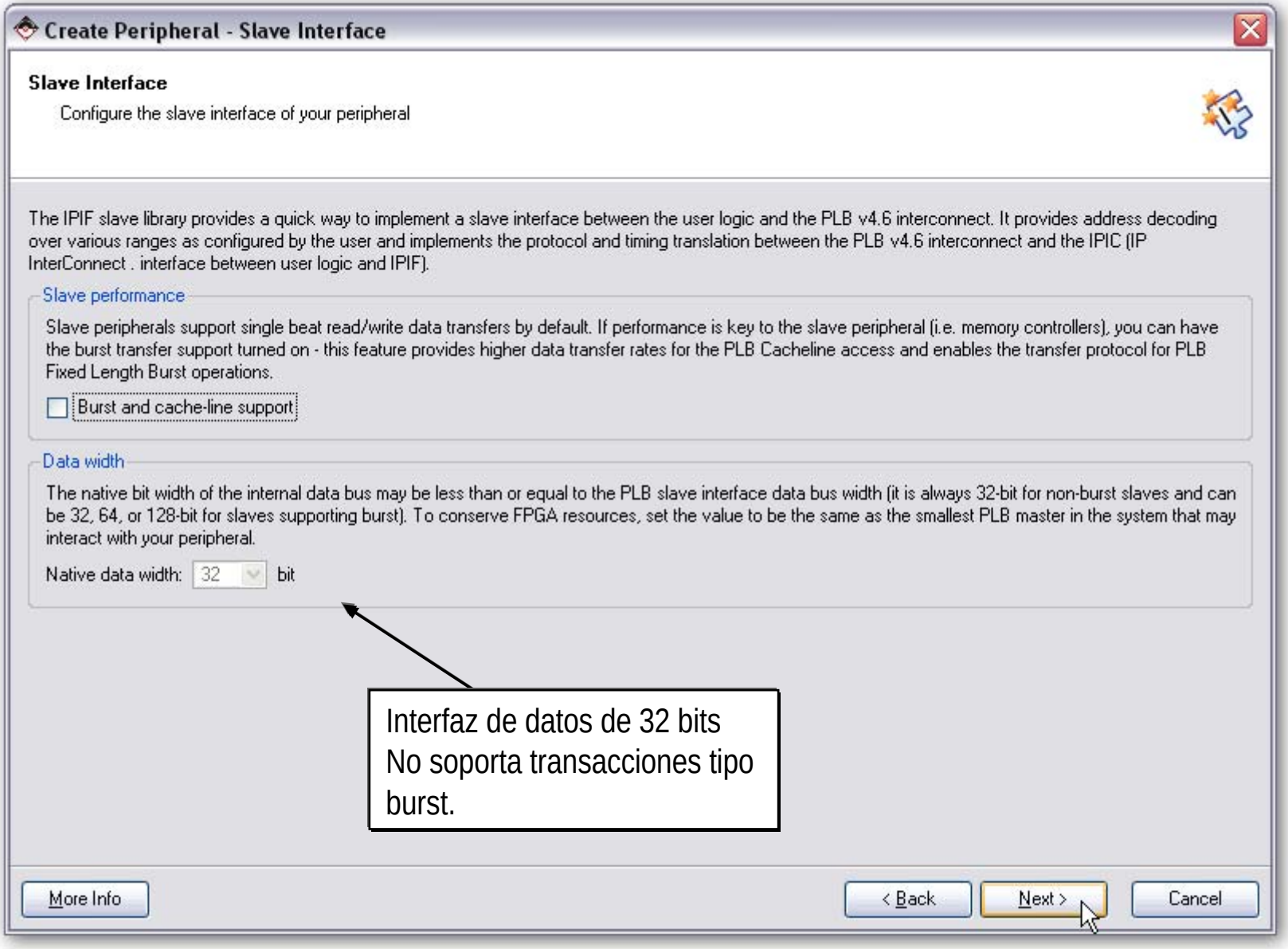
Master service and configuration

Typically required by complex peripherals like Ethernet and PCI for commanding data transfers between regions (PLB master interface will be included if master service selected).

- [ ] User logic master

More Info | < Back | Next > | Cancel

Utilización de la aplicación Create Peripheral Wizard



Interfaz de datos de 32 bits
No soporta transacciones tipo burst.

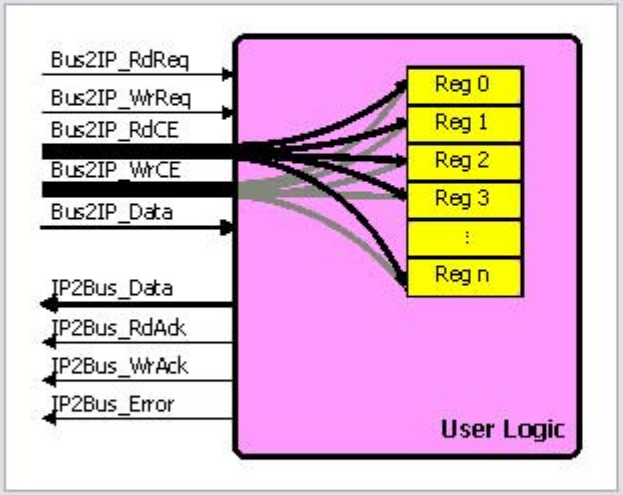
Utilización de la aplicación Create Peripheral Wizard

Create Peripheral - User S/W Register

User S/W Register

Configure the software accessible registers in your peripheral.

The user specific software accessible registers will be implemented in the user-logic module of your peripheral. Such registers are typically provided for software programs to control and to monitor the status of your user logic. These registers are addressable on the byte, half-word, word, double word or quad word boundaries depending on your design. An example logic for register read/write will be included in the user-logic module generated by the wizard tool for your reference.



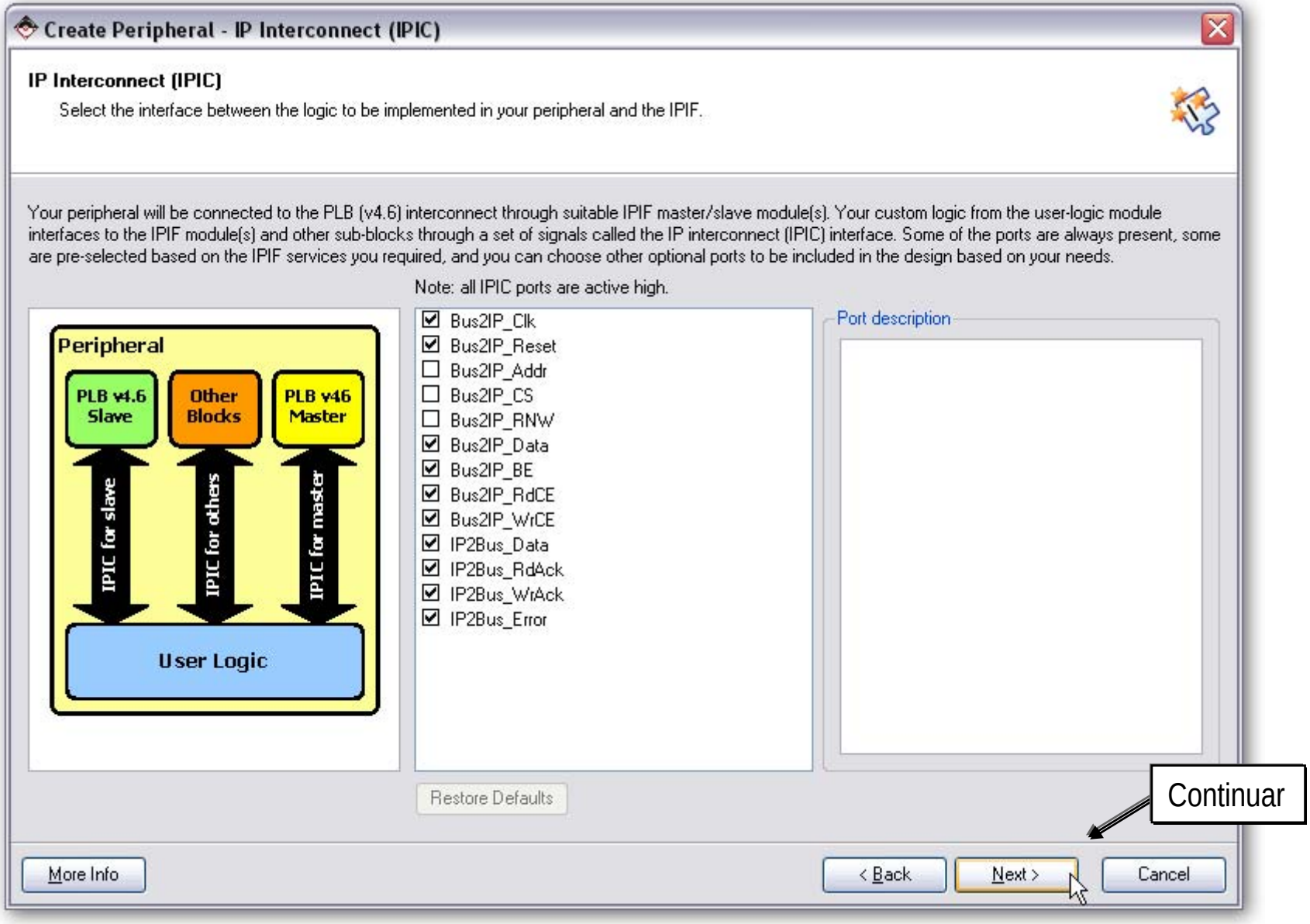
User logic software registers may take full advantage of the slave IPIF address-decoding service to generate CE decodes for all of the individual register of interest. The diagram on the left shows the simplest set of IPIC slave signals to read/write the registers.

Number of software accessible registers: 4 (1 to 4096)

4 registros

More Info | < Back | Next > | Cancel

Utilización de la aplicación Create Peripheral Wizard



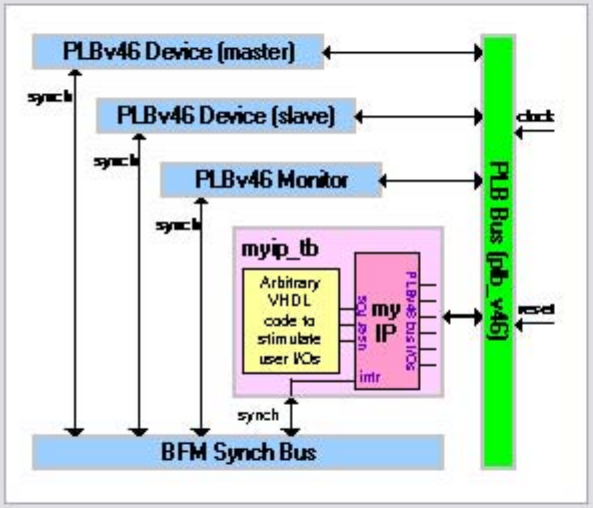
Utilización de la aplicación Create Peripheral Wizard

Create Peripheral - (OPTIONAL) Peripheral Simulation Support

(OPTIONAL) Peripheral Simulation Support

Generate optional files for simulation using Bus Functional Models (BFM).

The EDK provides a BFM simulation platform to help you simulate your peripheral. Indicate if you want this tool to generate the appropriate HDL and Bus Functional Language (BFL) stimulus file for the target bus.



☐ Generate BFM simulation platform for ModelSim-SE or ModelSim-PE

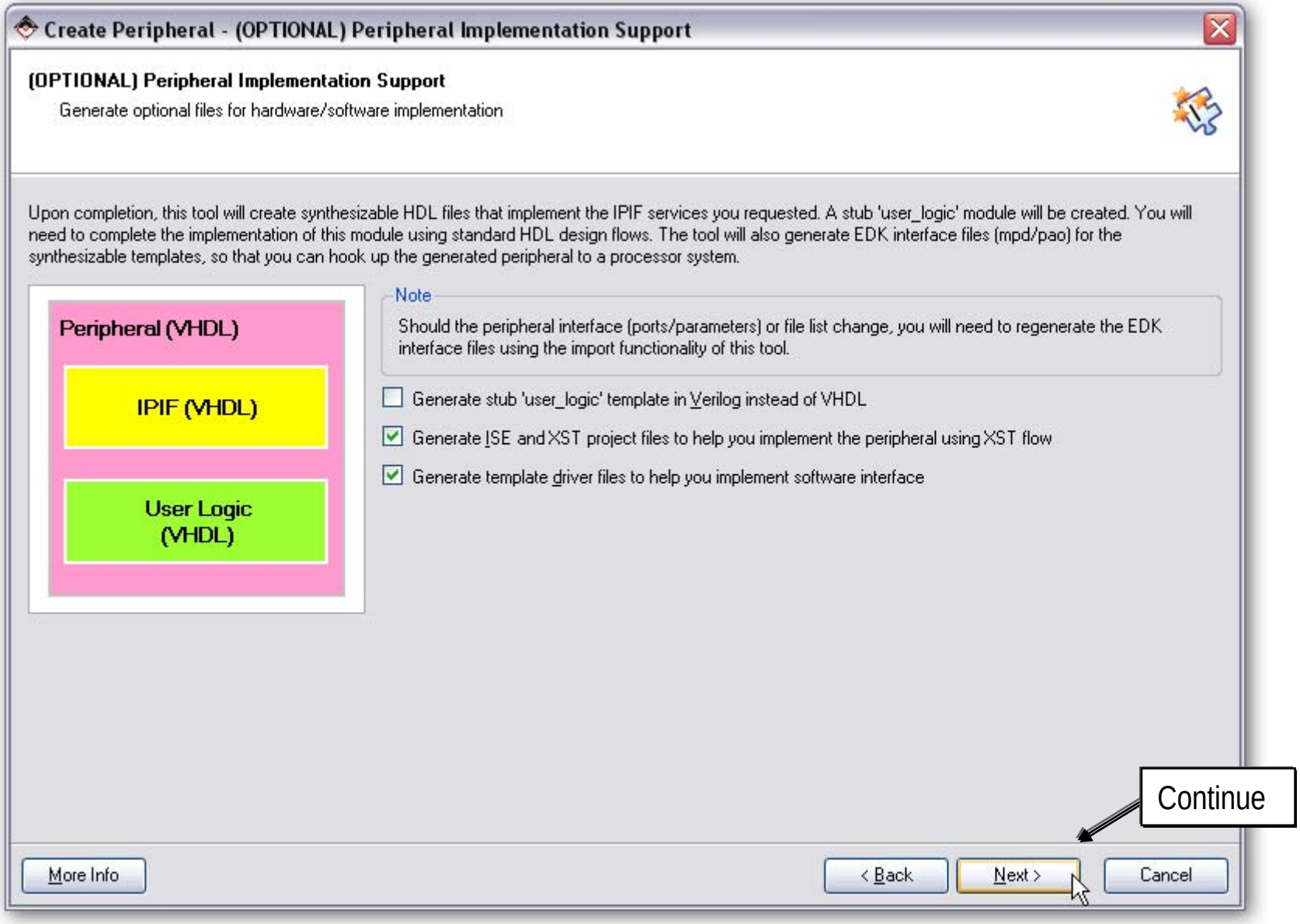
This feature requires that you have accepted the associated IBM license agreement and installed the BFM package. The link below shows how:

BFM Package Installation Instructions

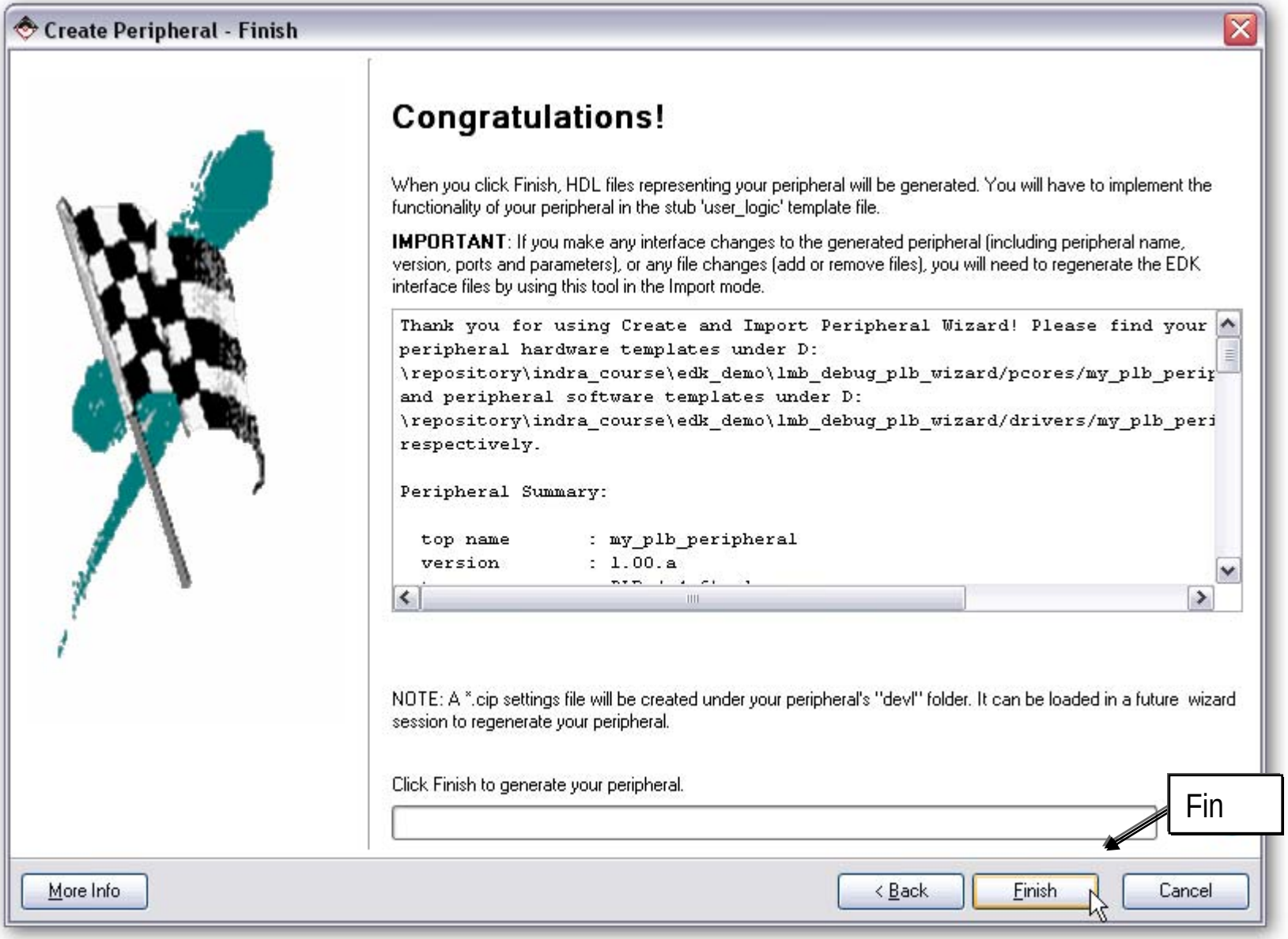
Continuar

More Info | < Back | Next > | Cancel

Utilización de la aplicación Create Peripheral Wizard



Utilización de la aplicación Create Peripheral Wizard



Utilización de la aplicación Create Peripheral Wizard

Xilinx Platform Studio - D:/repository/indra_course/edk_demo/1mb_debug_plb_wizard/system.xmp - [System Assembly View1]

File Edit View Project Hardware Software Device Configuration Debug Simulation Window Help

Project Information Area

Project | Applications | IP Catalog

Description | IP Versio

- EDK Install -- C:\Xilinx\10.1\ED...
  - Analog
  - Bus and Bridge
  - Clock, Reset and Interrupt
  - Communication High-Speed
  - Communication Low-Speed
  - DMA and Timer
  - Debug
  - General Purpose IO
  - Interprocessor Communication
  - Memory and Memory Controller
  - PCI
  - Peripheral Controller
  - Processor
  - Utility
- Project Local pcores -- D:\repositor...
  - USER
    - MY_PLB_PERIPHERAL 1.00.a

Processor Support:PowerPC,MicroBlaze

Status:DEVELOPMENT

Directory:D:\repository\indra_course\edk_demo\1mb_debug_plb_wizard\pcores\my_plb_peripheral_v1_00_a\data\

Bus Interfaces | Ports | Addresses | BIF Filters

Name	Bus Connection	IP Type	IP Version
microblaze_0		microblaze	7.10.c
mb_plb		plb_v46	1.03.a
xps_bram_if_cntlr_0		xps_bram_if_cntlr	1.00.a
PORTA	xps_bram_if_cntlr_0_PORTA		
SPLB	mb_plb		
bram_block_0		bram_block	1.00.a
PORTB	xps_bram_if_cntlr_0_PORTA		
PORTA	No Connection		
debug_module		mdm	1.00.c
RS232		xps_uartlite	1.00.a
SPLB	mb_plb		
clock_generator_0		clock_generator	2.01.a
proc_sys_reset_0		proc_sys_reset	2.00.a

Console Window: Output | Warning | Error

El periférico aparece disponible como un pcore en el proyecto EDK

Utilización de la aplicación Create Peripheral Wizard

Xilinx Platform Studio - D:/repository/indra_course/edk_demo/1mb_debug_plb_wizard/system.xmp - [System Assembly View1]

File Edit View Project Hardware Software Device Configuration Debug Simulation Window Help

Project Information Area: Project | Applications | IP Catalog

- Communication High-Speed
- Communication Low-Speed
- DMA and Timer
- Debug
- General Purpose IO
- Interprocessor Communication
- Memory and Memory Controller
- PCI
- Peripheral Controller
- Processor
- Utility
- Project Local pcores -- D:\repositor...
  - USER
    - MY_PLB_PERIPHERAL 1.00.a

Bus Interfaces | Ports | Addresses | BIF Filters

Name	Bus Connection	IP Type	IP Version
microblaze_0		microblaze	7.10.c
mb_plb		plb_v46	1.03.a
xps_bram_if_cntlr_0		xps_bram_if_cntlr	1.00.a
PORTA	xps_bram_if_cntlr_0_PORTA		
SPLB	mb_plb		
bram_block_0		bram_block	1.00.a
PORTB	xps_bram_if_cntlr_0_PORTA		
PORTA	No Connection		
debug_module		mdm	1.00.c
my_plb_peripheral_0		my_plb_peripheral	1.00.a
SPLB	mb_plb		
RS232		xps_uartlite	1.00.a
SPLB	mb_plb		
clock_generator_0		clock_generator	2.01.a
proc_sys_reset_0		proc_sys_reset	2.00.a

System Assembly View

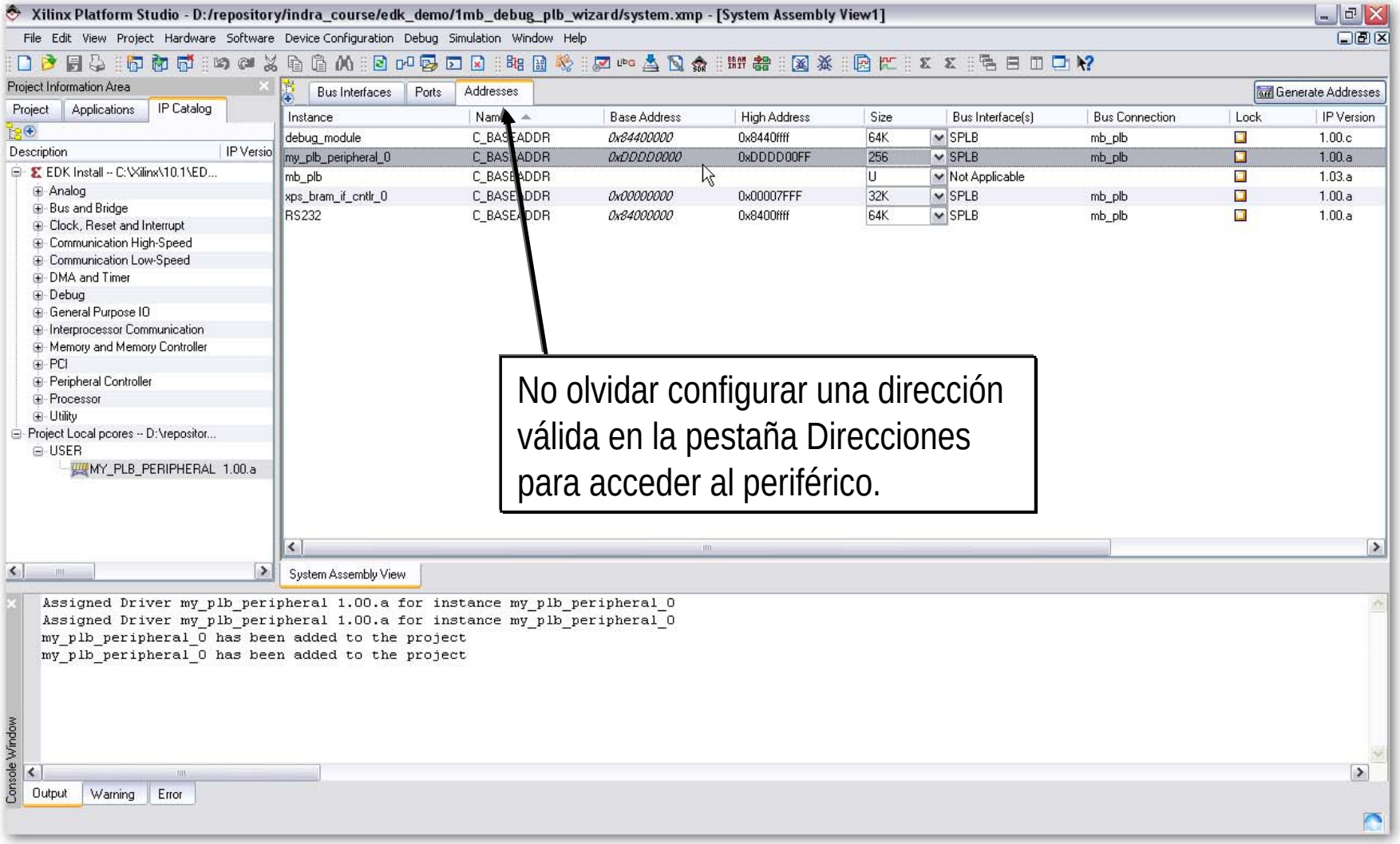
Console Window: Output | Warning | Error

```
Assigned Driver my_plb_peripheral 1.00.a for instance my_plb_peripheral_0
Assigned Driver my_plb_peripheral 1.00.a for instance my_plb_peripheral_0
my_plb_peripheral_0 has been added to the project
my_plb_peripheral_0 has been added to the project
```

Conectarlo al bus PLB haciendo click aquí

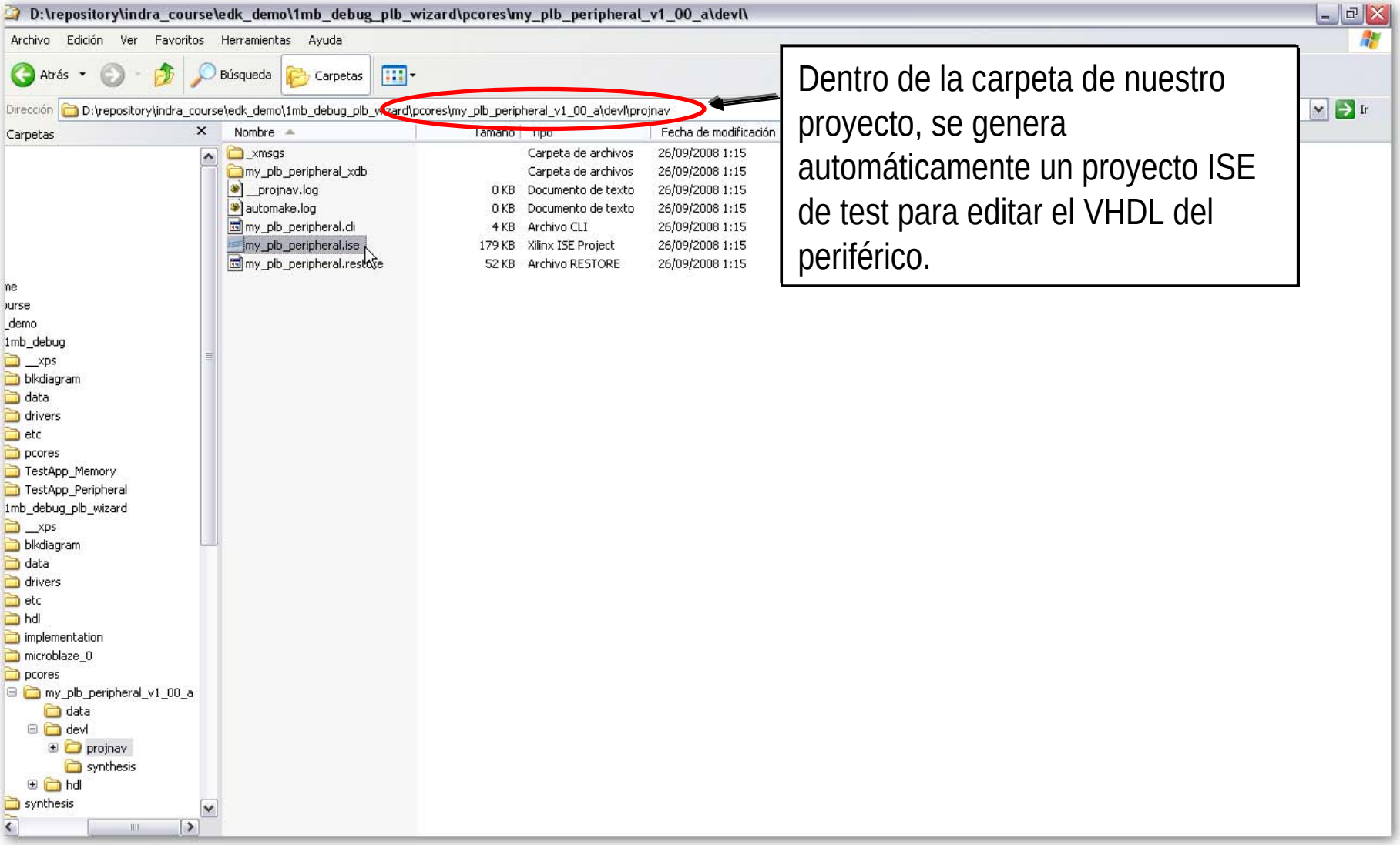
Para añadirlo arrastrar y soltar desde el IPCatalog sobre la ventana "System Assembly View ".

Utilización de la aplicación Create Peripheral Wizard



No olvidar configurar una dirección válida en la pestaña Direcciones para acceder al periférico.

Utilización de la aplicación Create Peripheral Wizard



Dentro de la carpeta de nuestro proyecto, se genera automáticamente un proyecto ISE de test para editar el VHDL del periférico.

Utilización de la aplicación Create Peripheral Wizard

El proyecto ISE tiene:
- Una entity top con el nombre dado al periférico
-Varios ficheros con el interfaz de bus
-User logic: template con el código VHDL del usuario

Xilinx - ISE - D:\repository\indra_course\edk_demo\1mb_d

File Edit View Project Source Process Window Help

Sources

Sources for: Implementation Number of: LUTs

my_plb_peripheral
xc3s1000-5fg456
my_plb_peripheral - IMP (D:/repository/indra_course/ed
PLBV46_SLAVE_SINGLE_I - plbv46_slave_single - im
USER_LOGIC_I - user_logic - IMP (D:/repository/indra_

USER_LOGIC_I - user_logic - IMP (D:/repository/indra_course/edk_demo/1mb_debug_plb_wizard/pcores/my_plb_peripheral_v1_00_a/hdl/vhdl/user_logic.vhd)

Sources Files Snapshots Libraries

Processes

Processes for: USER_LOGIC_I - user_logic - IMP

Add Existing Source
Create New Source
Design Utilities
Check Syntax

Processes

```
1
3
4
5
6  -- ** Copyright (c) 1995-2008 Xilinx, Inc.  All rights reserved.          **
7  -- **                                                                     **
8  -- ** Xilinx, Inc.                                                        **
9  -- ** XILINX IS PROVIDING THIS DESIGN, CODE, OR INFORMATION "AS IS"       **
11 -- ** SOLUTIONS FOR XILINX DEVICES.  BY PROVIDING THIS DESIGN, CODE,      **
12 -- ** OR INFORMATION AS ONE POSSIBLE IMPLEMENTATION OF THIS FEATURE,      **
13 -- ** APPLICATION OR STANDARD, XILINX IS MAKING NO REPRESENTATION         **
14 -- ** THAT THIS IMPLEMENTATION IS FREE FROM ANY CLAIMS OF INFRINGEMENT,   **
15 -- ** AND YOU ARE RESPONSIBLE FOR OBTAINING ANY RIGHTS YOU MAY REQUIRE    **
16 -- ** FOR YOUR IMPLEMENTATION.  XILINX EXPRESSLY DISCLAIMS ANY           **
17 -- ** WARRANTY WHATSOEVER WITH RESPECT TO THE ADEQUACY OF THE             **
18 -- ** IMPLEMENTATION, INCLUDING BUT NOT LIMITED TO ANY WARRANTIES OR      **
19 -- ** REPRESENTATIONS THAT THIS IMPLEMENTATION IS FREE FROM CLAIMS OF     **
20 -- ** INFRINGEMENT, IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS     **
21 -- ** FOR A PARTICULAR PURPOSE.                                            **
22 -- **                                                                     **
23 -- *********************************************************************
24 --
25 ------------------------------------------------------------------------
26 -- Filename:          user_logic.vhd
27 -- Version:           1.00.a
```

user_logic.vhd

Started : "Launching Design Summary".

Started : "Launching ISE Text Editor to edit user_logic.vhd".

Transcript

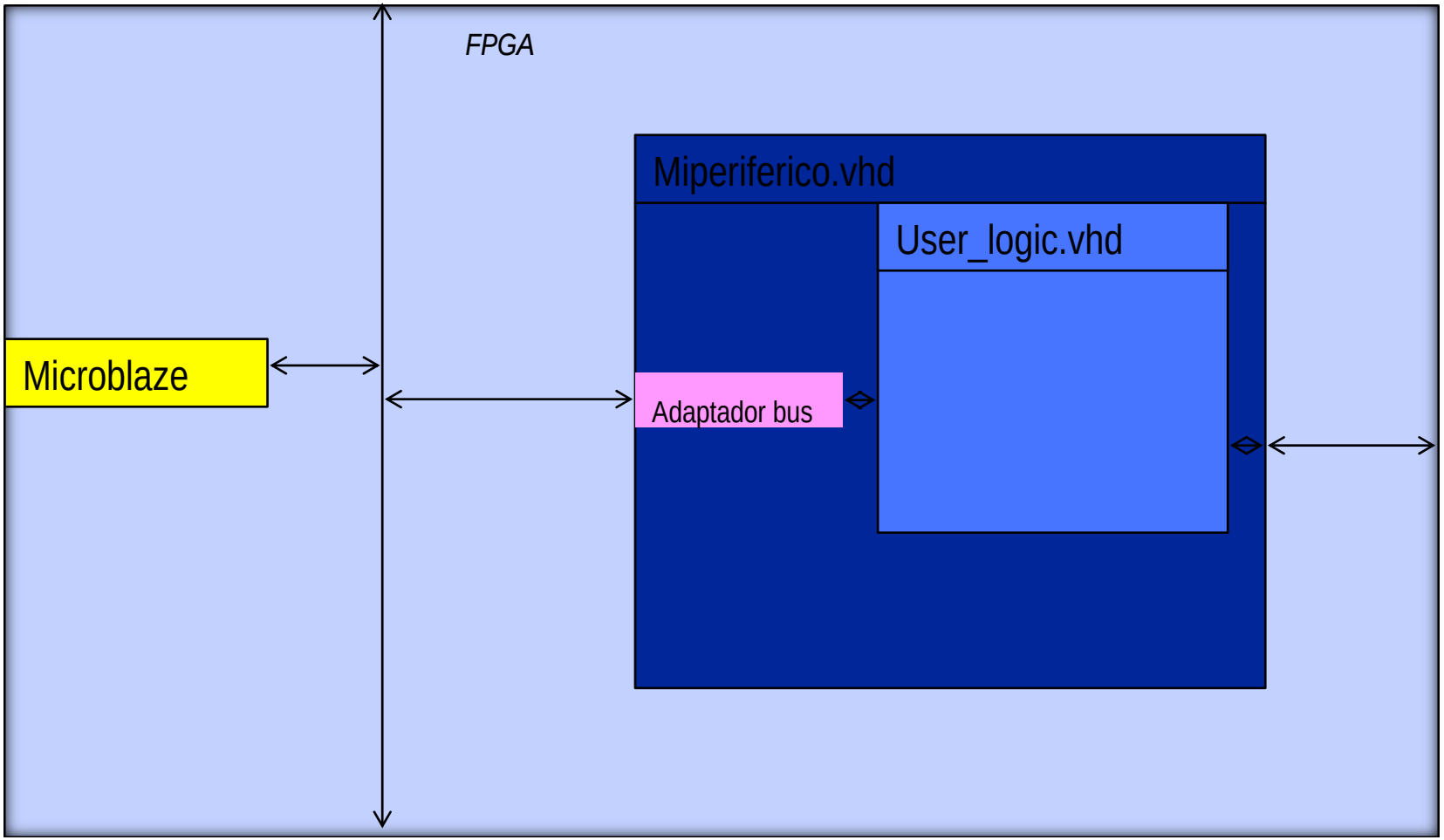
Console Errors Warnings Tcl Shell Find in Files

CAPS NUM SCRL Ln 1 Col 1 VHDL

Utilización de la aplicación Create Peripheral Wizard

- Modificar el periférico:
  - Abrir el proyecto ISE
  - Integrar el código del periférico en el fichero user_logic.vhd
  - Actualizar los puertos de in/out en el fichero *.vhd para añadir todos los que queramos
  - Actualizar los puertos de in/out en el fichero ".mpd"

Utilización de la aplicación Create Peripheral Wizard



Fichero miperiferico.vhd

```
-- Librerías
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 mi_coprocesador_v1_00_a;
use mi_coprocesador_v1_00_a.user_logic;
```

Fichero miperiferico.vhd

Entity Genéricas

```
entity mi_coprocesador 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_BASEADDR              : std_logic_vector    := X"FFFFFFFF";
    C_HIGHADDR              : std_logic_vector    := X"00000000";
    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_FAMILY                : string              := "virtex5"
    -- DO NOT EDIT ABOVE THIS LINE ---------------------
  );
```

Fichero miperiferico.vhd

Entity puertos

```
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
  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);
  --------------------- y unas cuantas más
);
```

Fichero miperiferico.vhd

Arquitectura

```
architecture IMP of mi_coprocesador is

 -------------- Definición de ctes para protocolo bus
-------------------------Señales de bus
 -- 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)

……..
```

Fichero miperiferico.vhd

```
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,
   ……….
  )
  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,
……..));
```

Fichero miperiferico.vhd

```
------------------------------------------
-- instantiate User Logic
------------------------------------------
USER_LOGIC_I : entity mi_coprocesador_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_DWIDTH                => USER_SLV_DWIDTH,
    C_NUM_REG                   => USER_NUM_REG
  )
  port map
  (
    -- MAP USER PORTS BELOW THIS LINE ------------------
    --USER ports mapped here
    -- MAP USER PORTS ABOVE THIS LINE ------------------

    Bus2IP_Clk                  => ipif_Bus2IP_Clk,
    Bus2IP_Reset                => ipif_Bus2IP_Reset,
    Bus2IP_Data                 => ipif_Bus2IP_Data,
    Bus2IP_BE                   => ipif_Bus2IP_BE,
    Bus2IP_RdCE                 => user_Bus2IP_RdCE,
……................................
  );
```

Fichero miperiferico.vhd

```
 ------------------------------------------
 -- connect internal signals
 ------------------------------------------
 ipif_IP2Bus_Data <= user_IP2Bus_Data;
 ipif_IP2Bus_WrAck <= user_IP2Bus_WrAck;
 ipif_IP2Bus_RdAck <= user_IP2Bus_RdAck;
 ipif_IP2Bus_Error <= user_IP2Bus_Error;

 user_Bus2IP_RdCE <= ipif_Bus2IP_RdCE(USER_CE_INDEX to
USER_CE_INDEX+USER_NUM_REG-1);
 user_Bus2IP_WrCE <= ipif_Bus2IP_WrCE(USER_CE_INDEX to
USER_CE_INDEX+USER_NUM_REG-1);

end IMP;
```

Fichero user_logic

```
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_DWIDTH                : integer           := 32;
    C_NUM_REG                   : integer           := 4
    -- DO NOT EDIT ABOVE THIS LINE ---------------------
  );
```

Fichero user_logic

```
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_Data   : in  std_logic_vector(0 to C_SLV_DWIDTH-1);
  Bus2IP_BE     : in  std_logic_vector(0 to C_SLV_DWIDTH/8-1);
  Bus2IP_RdCE   : in  std_logic_vector(0 to C_NUM_REG-1);
  Bus2IP_WrCE   : in  std_logic_vector(0 to C_NUM_REG-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 ---------------------
 );
```

Fichero user_logic

```
architecture IMP of user_logic is

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

  ---------------------------------------------
  -- Signals for user logic slave model s/w accessible register example
  ---------------------------------------------
  signal slv_reg0   : std_logic_vector(0 to C_SLV_DWIDTH-1);
  signal slv_reg1   : std_logic_vector(0 to C_SLV_DWIDTH-1);
  signal slv_reg2   : std_logic_vector(0 to C_SLV_DWIDTH-1);
  signal slv_reg3   : std_logic_vector(0 to C_SLV_DWIDTH-1);
  signal slv_reg_write_sel  : std_logic_vector(0 to 3);
  signal slv_reg_read_sel   : std_logic_vector(0 to 3);
  signal slv_ip2bus_data     : std_logic_vector(0 to C_SLV_DWIDTH-1);
  signal slv_read_ack     : std_logic;
  signal slv_write_ack      : std_logic;

begin
  --USER logic implementation added here ------------------------------------
  -- Example code to read/write user logic slave model s/w accessible registers --
```

Fichero user_logic

```
-- Each bit of the Bus2IP_WrCE/Bus2IP_RdCE signals is configured to correspond
 -- to one software accessible register by the top level template. For example,
 -- if you have four 32 bit software accessible registers in the user logic,
 -- you are basically operating on the following memory mapped registers:
 --
 --    Bus2IP_WrCE/Bus2IP_RdCE   Memory Mapped Register
 --                  "1000"   C_BASEADDR + 0x0
 --                  "0100"   C_BASEADDR + 0x4
 --                  "0010"   C_BASEADDR + 0x8
 --                  "0001"   C_BASEADDR + 0xC
 --
```

Fichero user_logic

```
slv_reg_write_sel <= Bus2IP_WrCE(0 to 3);
 slv_reg_read_sel  <= Bus2IP_RdCE(0 to 3);
 slv_write_ack     <= Bus2IP_WrCE(0) or Bus2IP_WrCE(1) or Bus2IP_WrCE(2) or Bus2IP_WrCE(3);
 slv_read_ack      <= Bus2IP_RdCE(0) or Bus2IP_RdCE(1) or Bus2IP_RdCE(2) or Bus2IP_RdCE(3);
```

Fichero

```
-- implement slave model software accessible register(s)
 SLAVE_REG_WRITE_PROC : process( Bus2IP_Clk ) is
 begin

  if Bus2IP_Clk'event and Bus2IP_Clk = '1' then
   if Bus2IP_Reset = '1' then
    slv_reg0 <= (others => '0');
    slv_reg1 <= (others => '0');
    slv_reg2 <= (others => '0');
    slv_reg3 <= (others => '0');
   else
    case slv_reg_write_sel is
     when "1000" =>
      for byte_index in 0 to (C_SLV_DWIDTH/8)-1 loop
       if ( Bus2IP_BE(byte_index) = '1' ) then
        slv_reg0(byte_index*8 to byte_index*8+7) <= Bus2IP_Data(byte_index*8 to byte_index*8+7);
       end if;
      end loop;
     when "0100" =>
………                                   Y lo mismo para reg1, reg2 y reg 3. Puede añadirse más funcionalidad a los registros
    end case;
   end if;
  end if;
 end process SLAVE_REG_WRITE_PROC;
```

Fichero

```
 SLAVE_REG_READ_PROC : process( slv_reg_read_sel, slv_reg0, slv_reg1, slv_reg2,
slv_reg3 ) is
  begin

    case slv_reg_read_sel is
      when "1000" => slv_ip2bus_data <= slv_reg0;
      when "0100" => slv_ip2bus_data <= slv_reg1;
      when "0010" => slv_ip2bus_data <= slv_reg2;
      when "0001" => slv_ip2bus_data <= slv_reg3;
      when others => slv_ip2bus_data <= (others => '0');
    end case;

  end process SLAVE_REG_READ_PROC;


IP2Bus_Data  <= slv_ip2bus_data when slv_read_ack = '1' else
                (others => '0');


 IP2Bus_WrAck <= slv_write_ack;
 IP2Bus_RdAck <= slv_read_ack;
 IP2Bus_Error <= '0';
-----Añadir lógica del periférico
end IMP;
```

Utilización de la aplicación Create Peripheral Wizard

- Propagar las modificaciones: ISE -> EDK
  - Cambios dentro del periférico no necesitan ninguna actualización
  - Cambios en el interface del proyecto ISE sólo se propagan si reimportamos el periférico o si modificamos manualmente los ficheros de configuración (dentro de "my_peripheral/data/")
    - ✓ Si añadimos...
      - ➢ *Un nuevo port a la entidad top => Modificar el fichero .mpd*

        Por ejemplo si añadimos 4 switches, 4 leds y un push-button pondremos
        - **PORT switches_pin = "", DIR = I, VEC = [0:3], ENDIAN = LITTLE**
        - **PORT leds_pin="", DIR = O, VEC = [0:3], ENDIAN = LITTLE**
        - **PORT mibutton="", DIR = I,**
      - ➢ *Un nuevo fichero en el proyecto ISE => Modificar el fichero .pao*

Utilización de la aplicación Create Peripheral Wizard

```
1   ##########################################################################
2   ## Filename:          D:/repository/indra_course/labs_pablo_14-10-2008/Davi
3   ## Description:       Peripheral Analysis Order
4   ## Date:              Mon Oct 20 12:08:49 2008 (by Create and Import Periph
5   ##########################################################################
6
7   lib proc_common_v2_00_a proc_common_pkg vhdl
8   lib proc_common_v2_00_a ipif_pkg vhdl
9   lib proc_common_v2_00_a or_muxcy vhdl
10  lib proc_common_v2_00_a or_gate128 vhdl
11  lib proc_common_v2_00_a family_support vhdl
12  lib proc_common_v2_00_a pselect_f vhdl
13  lib proc_common_v2_00_a counter_f vhdl
14  lib plbv46_slave_single_v1_00_a plb_address_decoder vhdl
15  lib plbv46_slave_single_v1_00_a plb_slave_attachment vhdl
16  lib plbv46_slave_single_v1_00_a plbv46_slave_single vhdl
17  lib my_plb_peripheral_v1_00_a my_modules vhdl
18  lib my_plb_peripheral_v1_00_a user_logic vhdl
19  lib my_plb_peripheral_v1_00_a my_plb_peripheral vhdl
20
```

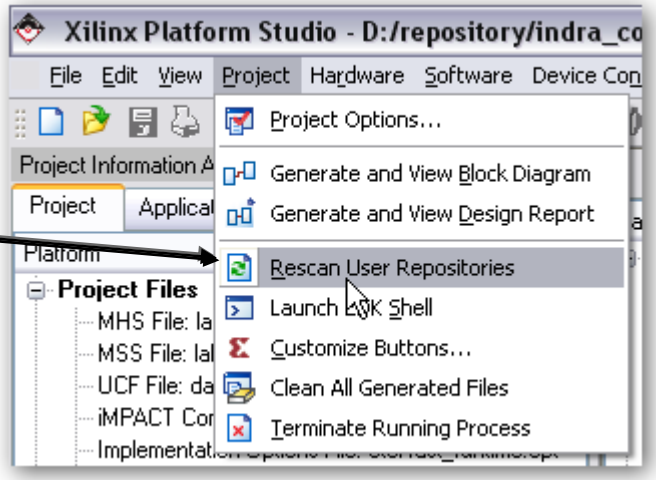
my_plb_peripheral_v2_1_0.mpd | my_plb_peripheral_v2_1_0.pao | my_modules.vhd

Si añadimos nuevos ficheros VHDL hay que indicárselo al EDK

Utilización de la aplicación Create Peripheral Wizard

- Abrir el proyecto EDK

Si EDK estaba ya abierto seleccionar :
Project -> Rescan User Repositories
Para actualizar los datos



Utilización de la aplicación Create Peripheral Wizard

Bus Interfaces | Ports | Addresses

Name	Net	Direction	Range
External Ports			
select_sw_not_counter_in_pin	select_sw_not_counter_in	I	
reset_counter_in_pin	reset_counter_in	I	
load_reg_switches_in_pin	load_reg_switches_in	I	
switches_in_pin	switches_in	I	[0:3]
leds_from_reg_out_pin	leds_from_reg_out	O	[0:3]
leds_from_sw_or_counter_out...	leds_from_sw_or_counter_out	O	[0:3]
sys_rst_pin	sys_rst_s	I	
sys_clk_pin	dcm_clk_s	I	
fpga_0_RS232_TX_pin	fpga_0_RS232_TX	O	
fpga_0_RS232_RX_pin	fpga_0_RS232_RX	I	
microblaze_0			
debug_module			
my_plb_peripheral_0			
leds_from_sw_or_counter_out	leds_from_sw_or_counter_out	O	[0:3]
leds_from_reg_out	leds_from_reg_out	O	[0:3]
switches_in	switches_in	I	[0:3]
load_reg_switches_in	load_reg_switches_in	I	
reset_counter_in	reset_counter_in	I	
select_sw_not_counter_in	select_sw_not_counter_in	I	
RS232			

System Assembly View

Los nuevos puertos deberían aparecer. Conectarlos como puertos externos

Utilización de la aplicación Create Peripheral Wizard

```
##############################################################################
## This system.ucf file is generated by Base System Builder based on the
## settings in the selected Xilinx Board Definition file. Please add other
## user constraints to this file based on customer design specifications.
##############################################################################

Net sys_clk_pin LOC=P8;
#push buttons placa superior
#NET SW2 LOC=E11;
#NET SW3 LOC=A13;
Net sys_rst_pin LOC=E11;
## System level constraints
Net sys_clk_pin TNM_NET = sys_clk_pin;
TIMESPEC TS_sys_clk_pin = PERIOD sys_clk_pin 20000 ps;
Net sys_rst_pin TIG;

## IO Devices constraints

#### Module RS232 constraints

Net fpga_0_RS232_RX_pin LOC=G5;
Net fpga_0_RS232_TX_pin LOC=J2;

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

#switches
NET switches_in_pin<0> LOC=K4;
NET switches_in_pin<1> LOC=K3;
NET switches_in_pin<2> LOC=K2;
NET switches_in_pin<3> LOC=J4;
##################################################

# Push button placa extendida
#NET pushb1 LOC=H4;
#NET pushb2 LOC=L5;
#NET pushb3 LOC=N2;
#NET pushb4 LOC=M3;

NET load_reg_switches_in_pin LOC=H4;
NET reset_counter_in_pin LOC=M3;
```

Update the ".UCF"

```
##################################################

#switches placa extendida
#NET DIPSW<1> LOC=P12;
#NET DIPSW<2> LOC=J1;
#NET DIPSW<3> LOC=H1;
#NET DIPSW<4> LOC=H3;
#NET DIPSW<5> LOC=G2;
#NET DIPSW<6> LOC=K15;
#NET DIPSW<7> LOC=K16;
#NET DIPSW<8> LOC=F15;

NET select_sw_not_counter_in_pin LOC=P12;

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

#barra de leds placa extendida
#NET leds<1> LOC=L5;
#NET leds<2> LOC=N2;
#NET leds<3> LOC=M3;
#NET leds<4> LOC=N1;
#NET leds<5> LOC=T13;
#NET leds<6> LOC=L15;
#NET leds<7> LOC=J13;
#NET leds<8> LOC=H15;
#NET leds<9> LOC=J16;
#NET leds<10> LOC=J14;


#leds
NET leds_from_sw_or_counter_out_pin<0> LOC=L5;
NET leds_from_sw_or_counter_out_pin<1> LOC=N2;
NET leds_from_sw_or_counter_out_pin<2> LOC=M3;
NET leds_from_sw_or_counter_out_pin<3> LOC=N1;

#reg leds
NET leds_from_reg_out_pin<0> LOC=J13;
NET leds_from_reg_out_pin<1> LOC=H15;
NET leds_from_reg_out_pin<2> LOC=J16;
NET leds_from_reg_out_pin<3> LOC=J14;
```

Utilización de la aplicación Create Peripheral Wizard

- SW:
  - Incluir la aplicación sw: “TestApp_Peripheral”
  - Como ayuda para generar la aplicación disponemos, dentro de la carpeta del proyecto, en el directorio drivers, de una carpeta para cada periférico con los siguientes ficheros:
    - Myperipheral_name/Src/makefile
    - Myperipheral_name/Src/Myperipheral_name.h
    - Myperipheral_name/Src/Myperipheral_name.c
    - Myperipheral_name/Src/Myperipheral_name_selftest.c

- **Myperipheral_name.h**

```
/**************************** Include Files ******************************/

#include "xbasic_types.h"
#include "xstatus.h"
#include "xio.h"

/************************** Constant Definitions **************************/


/**
 * User Logic Slave Space Offsets
 * -- SLV_REG0 : user logic slave module register 0
 * -- SLV_REG1 : user logic slave module register 1
 * -- SLV_REG2 : user logic slave module register 2
 * -- SLV_REG3 : user logic slave module register 3
 */
#define MY_PERIPHERAL_USER_SLV_SPACE_OFFSET (0x00000000)
#define MY_PERIPHERAL_SLV_REG0_OFFSET (MY_PERIPHERAL_USER_SLV_SPACE_OFFSET + 0x00000000)
#define MY_PERIPHERAL_SLV_REG1_OFFSET (MY_PERIPHERAL_USER_SLV_SPACE_OFFSET + 0x00000004)
#define MY_PERIPHERAL_SLV_REG2_OFFSET (MY_PERIPHERAL_USER_SLV_SPACE_OFFSET + 0x00000008)
#define MY_PERIPHERAL_SLV_REG3_OFFSET (MY_PERIPHERAL_USER_SLV_SPACE_OFFSET + 0x0000000C)
```

Utilización de la aplicación Create Peripheral Wizard

```
/**************** Macros (Inline Functions) Definitions *****************/

/** Escritura en un registro (32 bits)
 *
 *  @param   BaseAddress is the base address of the MY_PERIPHERAL device.
 * @param   RegOffset is the register offset from the base to write to.
 * @param   Data is the data written to the register.
 *
 * @return  None.
 **
 */
#define MY_PERIPHERAL_mWriteReg(BaseAddress, RegOffset, Data) \
            xil_io_out32((BaseAddress) + (RegOffset), (Xuint32)(Data))
```

Utilización de la aplicación Create Peripheral Wizard

```
/**
 *
 * Lectura en un registro del periférico (32 bits)
 *
 * @param   BaseAddress is the base address of the MY_PERIPHERAL device.
 * @param   RegOffset is the register offset from the base to write to.
 *
 * @return  Data is the data from the register.
 *
*/
#define MY_PERIPHERAL_mReadReg(BaseAddress, RegOffset) \
        xil_io_in32((BaseAddress) + (RegOffset))
```

Utilización de la aplicación Create Peripheral Wizard

```
/**
 * Lecturas y escrituras de 32 bits en los registros internos de my:peripheral
 *
 * @param   BaseAddress is the base address of the MY_PERIPHERAL device.
 * @param   RegOffset is the offset from the slave register to write to or read from.
 * @param   Value is the data written to the register.
 * @return  Data is the data from the user logic slave register.
 ** C-style signature:
 *                void MY_PERIPHERAL_mWriteSlaveRegn(Xuint32 BaseAddress, unsigned RegOffset, Xuint32 Value)
 *                Xuint32 MY_PERIPHERAL_mReadSlaveRegn(Xuint32 BaseAddress, unsigned RegOffset)
 */
#define MY_PERIPHERAL_mWriteSlaveReg0(BaseAddress, RegOffset, Value) \
                xil_io_out32((BaseAddress) + (MY_PERIPHERAL_SLV_REG0_OFFSET) + (RegOffset),
(Xuint32)(Value))
#define MY_PERIPHERAL_mReadSlaveReg0(BaseAddress, RegOffset) \
                xil_io_in32((BaseAddress) + (MY_PERIPHERAL_SLV_REG0_OFFSET) +
(RegOffset))

#define MY_PERIPHERAL_mWriteSlaveReg1(BaseAddress, RegOffset, Value) \
                xil_io_out32((BaseAddress) + (MY_PERIPHERAL_SLV_REG1_OFFSET) + (RegOffset), (Xuint32)(Value))
#define MY_PERIPHERAL_mReadSlaveReg1(BaseAddress, RegOffset) \
                xil_io_in32((BaseAddress) + (MY_PERIPHERAL_SLV_REG1_OFFSET) + (RegOffset))
```

- **Myperipheral_name.c**

```
/**************************** Include Files
******************************/

#include "my_peripheral.h"

/************************** Function Definitions
***************************/
```

• Myperipheral_name_selftest.c

```
* @param   baseaddr_p  is the base address of the MY_PERIPHERAL instance to be worked on.
*
* @return
*
*    - XST_SUCCESS   if all self-test code passed
*    - XST_FAILURE   if any self-test code failed
*

*/
XStatus MY_PERIPHERAL_SelfTest(void * baseaddr_p)
{
  int     Index;
  Xuint32 baseaddr;
  Xuint8  Reg8Value;
  Xuint16 Reg16Value;
  Xuint32 Reg32Value;
```

Utilización de la aplicación Create Peripheral Wizard

```
/*
 * Check and get the device address
 */
/*
 * Base Address maybe 0. Up to developer to uncomment line below.
XASSERT_NONVOID(baseaddr_p != XNULL);
 */
baseaddr = (Xuint32) baseaddr_p;

xil_printf("******************************\n\r");
xil_printf("* User Peripheral Self Test\n\r");
xil_printf("******************************\n\n\r");
```

Utilización de la aplicación Create Peripheral Wizard

```
xil_printf("   - write 1 to slave register 0 word 0\n\r");
 MY_PERIPHERAL_mWriteSlaveReg0(baseaddr, 0, 1);
 Reg32Value = MY_PERIPHERAL_mReadSlaveReg0(baseaddr, 0);
 xil_printf("   - read %d from register 0 word 0\n\r", Reg32Value);
 if ( Reg32Value != (Xuint32) 1 )
 {
   xil_printf("   - slave register 0 word 0 write/read failed\n\r");
   return XST_FAILURE;
 }
 xil_printf("   - write 2 to slave register 1 word 0\n\r");
 MY_PERIPHERAL_mWriteSlaveReg1(baseaddr, 0, 2);
 Reg32Value = MY_PERIPHERAL_mReadSlaveReg1(baseaddr, 0);
 xil_printf("   - read %d from register 1 word 0\n\r", Reg32Value);
 if ( Reg32Value != (Xuint32) 2 )
 {
   xil_printf("   - slave register 1 word 0 write/read failed\n\r");
   return XST_FAILURE;
```

Utilización de la aplicación Create Peripheral Wizard

Y así para todos los registros del periférico. Si todos se leen correctamente devuelve XST_SUCCESS