1. Adding Custom IP to an Embedded System Using AXI

Introduction

This lab guides you through the process of adding a custom peripheral to a processor system by using the Create and Import Peripheral Wizard.

Objectives

After completing this lab, you will be able to:

- Create a custom peripheral
- Add the custom peripheral to your design
- Add pin location constraints
- Generate the hardware bitstream

Procedure

This lab is separated into steps that consist of general overview statements that provide information on the detailed instructions that follow. Follow these detailed instructions to progress through the lab.

This lab comprises 4 primary steps: You will open the project, generate a peripheral template, create a peripheral, and, finally, add and connect the peripheral.

Design Description

You will extend the Lab 2 hardware design by creating and adding a AXIperipheral (refer to MYIP in **Figure 1**) to the system, and connecting it to the LEDs on the Atlys Board. You will use the Create and Import Peripheral Wizard of Xilinx Platform Studio (XPS) to generate the peripheral templates. You will complete the peripheral by adding LEDs interface logic in the templates. Next, you will connect the peripheral to the system and add pin location constraints to connect the LEDs controller peripheral to the on-board LEDs.

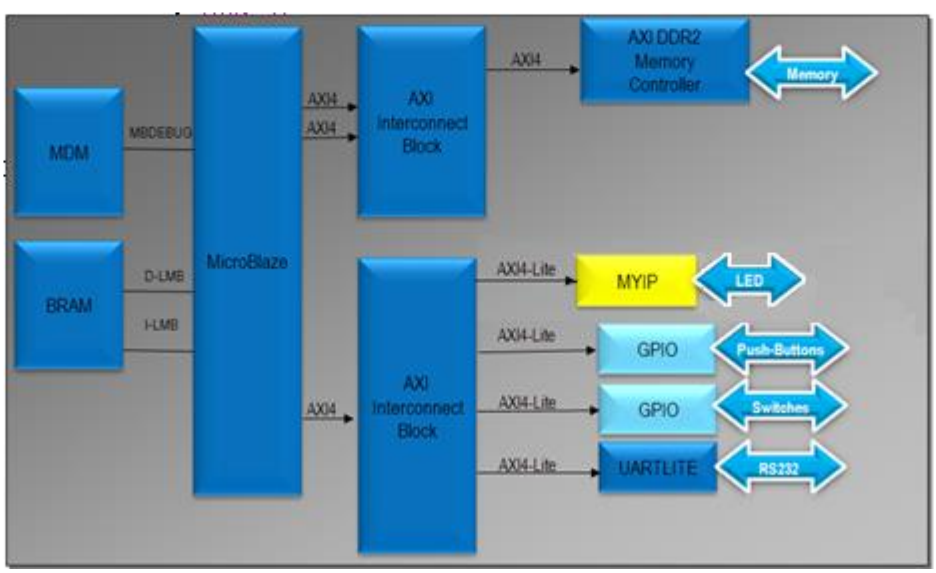

Figure 1. Design updated from previous lab

General Flow for this Lab

Step 1: Open the project → Step 2: Generate a peripheral template → Step 3: Create a peripheral → Step 4: Add and connect the peripheral

Opening the Project Step 1

1-1. Create a *lab3* folder and copy the contents of the *lab2* folder into the *lab3* folder if you wish to continue with the design you created in the previous lab, otherwise copy the *lab2* folder content from the *labsolution* folder into the *lab3* folder. Open the project in XPS.

1-1-1. If you wish to continue using the design that you created in Lab 2, create a *lab3* folder in the c*:\xup\embedded\labs* directory and copy the contents from *lab2* to *lab3*, otherwise copy the content of *lab2* folder from the *labsolution* folder.

1-1-2. Open XPS by clicking **Start > All Programs > Xilinx Design Tools > ISE Design Suite 14.2 > EDK > Xilinx Platform Studio.**

1-1-3. Select **Open project,** and browse to *C:\xup\embedded\labs\lab3.*

1-1-4. Click **system.xmp,** and click **Open** to open the project.

Generate a Peripheral Template Step 2

2-1. You will use the Create/Import Peripheral Wizard to create AXILite interface peripheral template.

2-1-1. In XPS, select **Hardware > Create or Import Peripheral…** to start the wizard.

2-1-2. Click **Next** to continue to the Create and Import Peripheral Wizard flow selection.

2-1-3. In the *Select Flow* panel, select **Create templates for a new peripheral** and click **Next.**

2-1-4. Click **next** with the default option **To an XPS project** selected.

2-1-5. Click **Next** and enter ***led_ip*** in the Name field, leave the default version number of 1.00.a, and click **Next**.

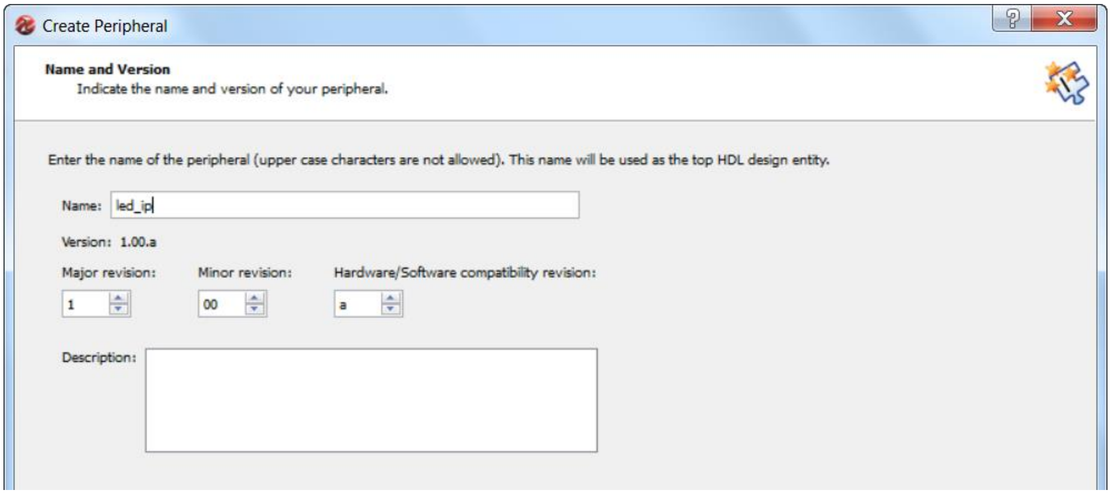

Figure 2. Provide Core Name and Version Number

2-1-6. Select **AXI-Lite** interface and click **Next.**

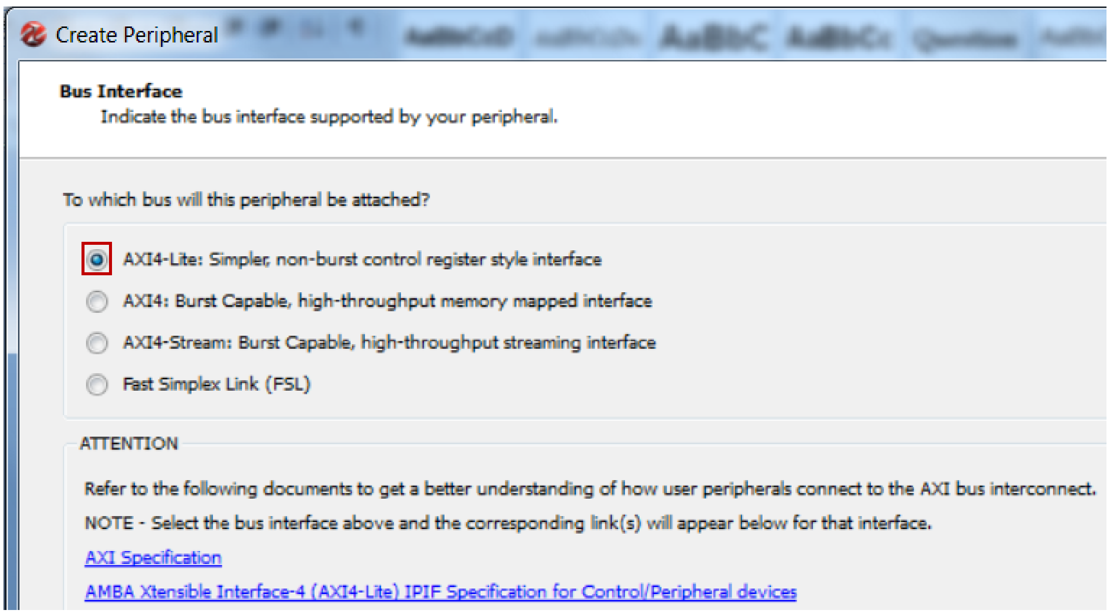

Figure 3. Select the AXI-Lite interface

2-2. Continuing with the wizard, select User Logic S/W Register support. Select only one software accessible register of 32-bit width. Generate template driver files. Browse to the C:\xup\embedded\labs\lab3 directory and ensure the structure.

2-2-1. In the IPIF Services panel, deselect **Include data phase timer**, leaving **User logic software register** selected, and click **Next**.

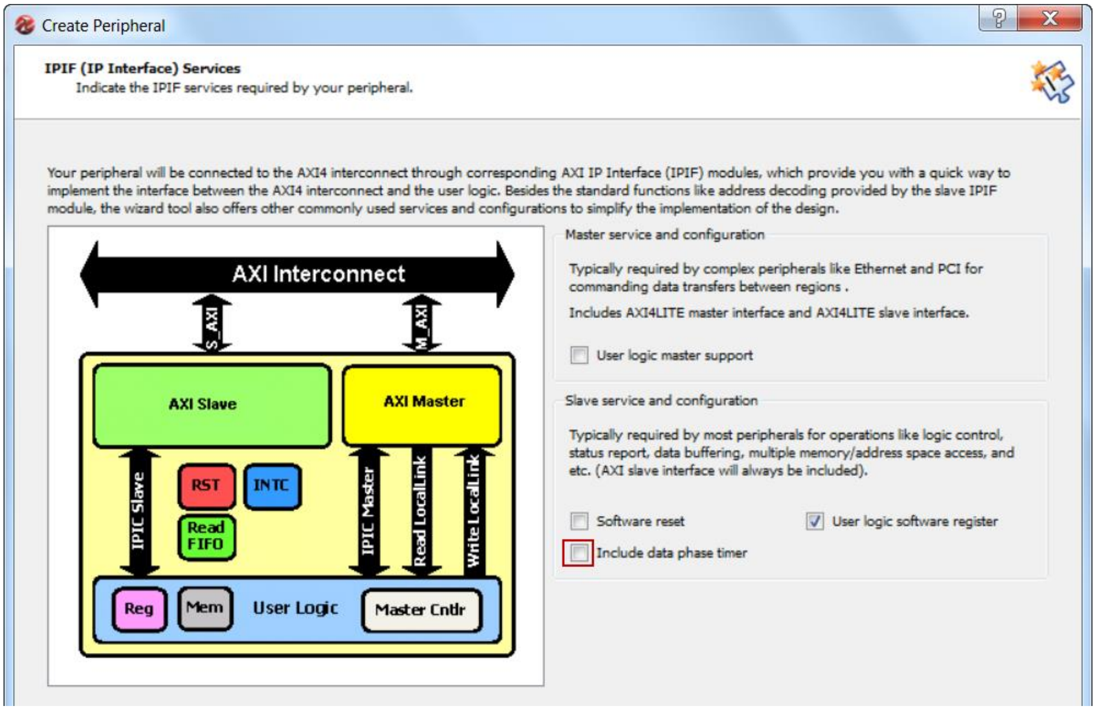

Figure 4. IPIF Services Dialogue Box

2-2-2. Click **Next,** accepting the default number of registers to be **1**.

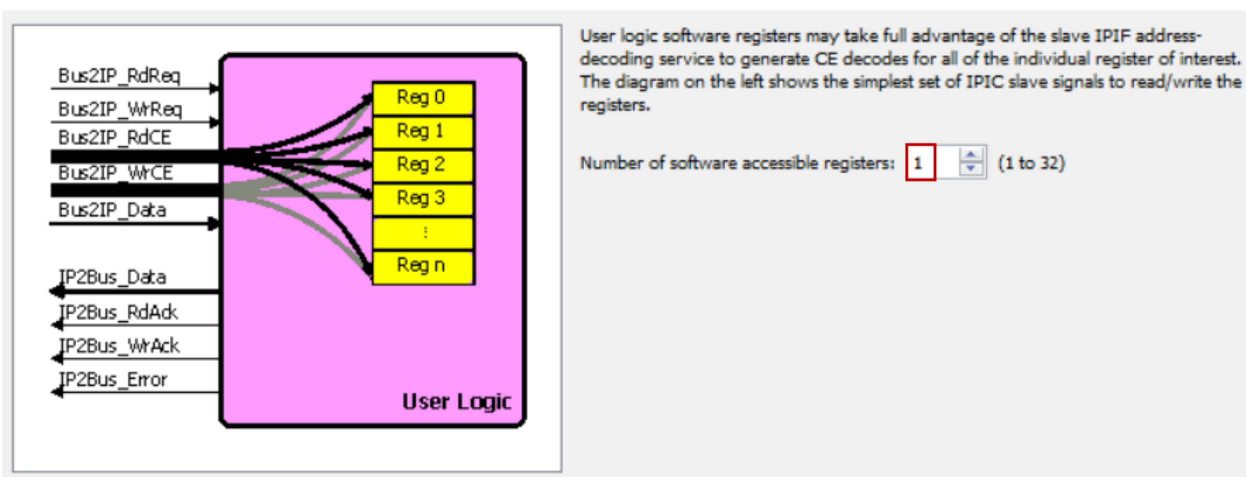

Figure 5. User SW Registers

2-2-3. Scroll through the **IP Interconnect (IPIC)** panel, which displays the default IPIC signals that are available for the user logic based on the previous selection. Notice that Bus2IP_Addr, Bus2IP_CS, Bus2IP_RNW are not checked since the custom peripheral does not have any memory support. Click **Next.**

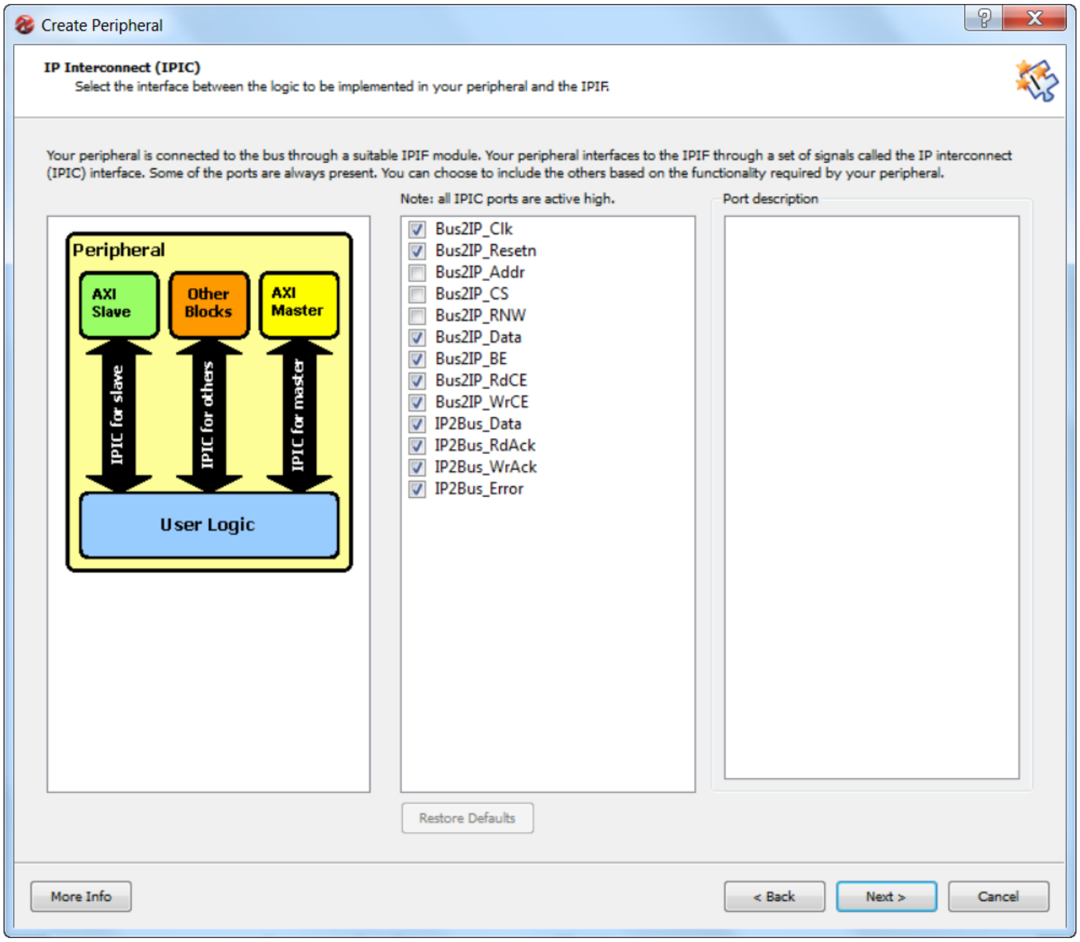

Figure 6. IP Interconnect (IPIC) Dialog Box

2-2-4. In the **(OPTIONAL) Peripheral Simulation Support** panel, leave **Generate BFM simulation platform** unchecked, and click **Next.**

2-2-5. In the **(OPTIONAL) Peripheral Implementation Options** panel, click **Generate template driver files to help you to implement software interface,** leaving others unchecked.

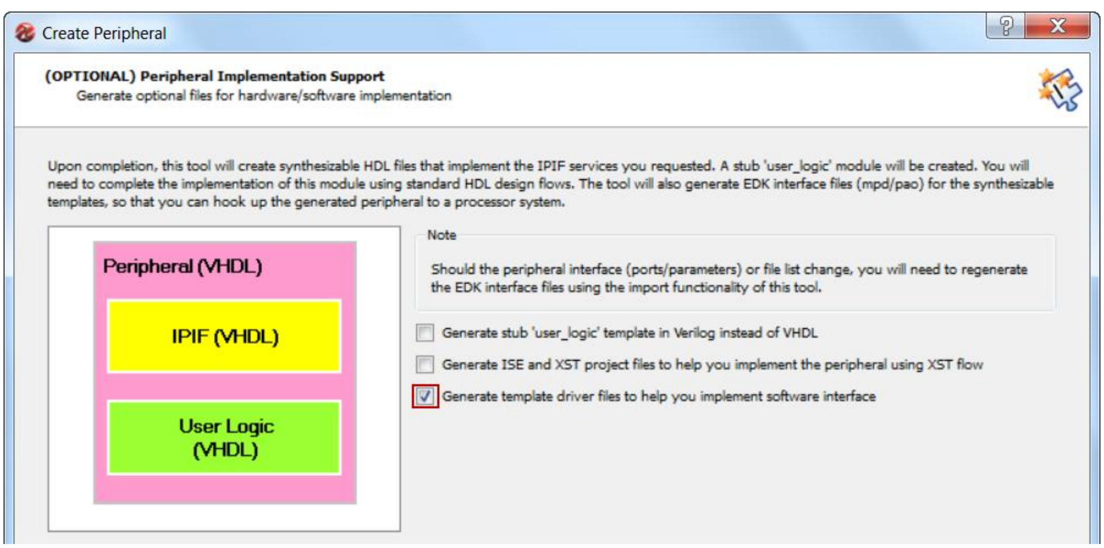

Figure 7. Peripheral Implementation Options dialog box

2-2-6. Click **Next**, and you will see the summary information panel.

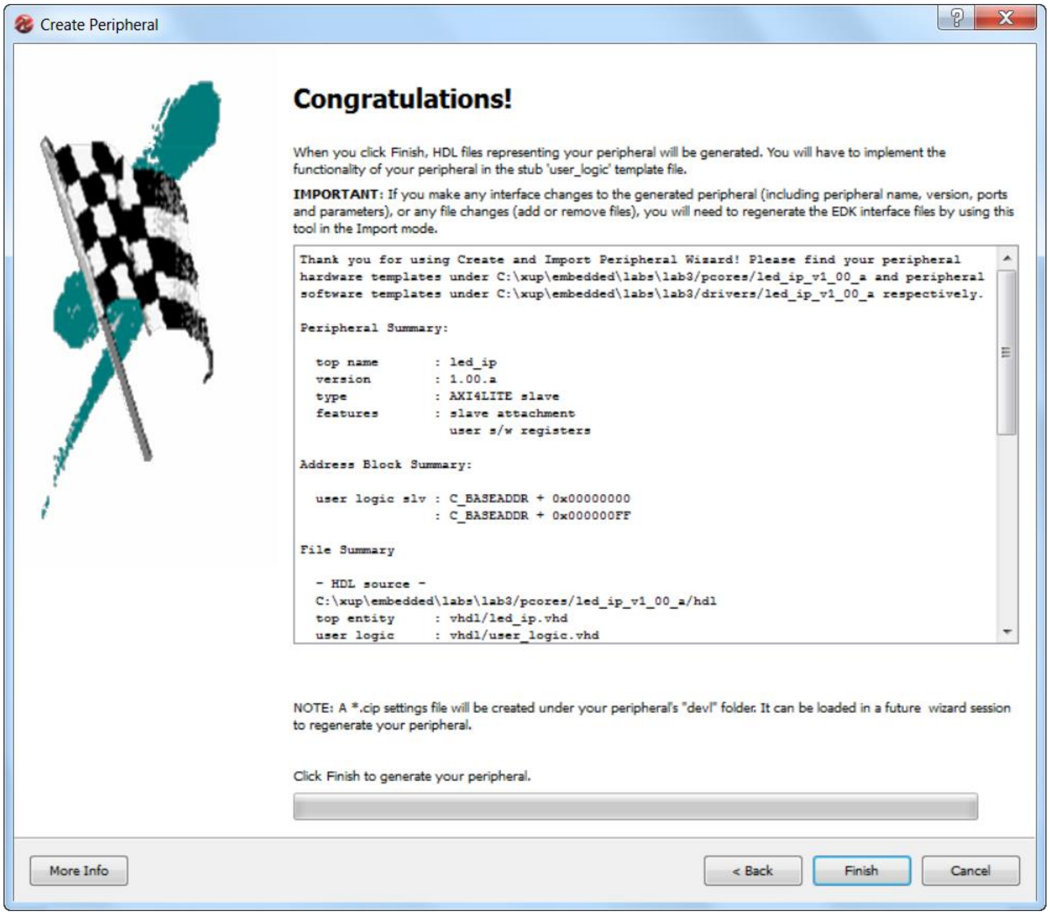

Figure 8. Congratulations dialog box

2-2-7. Click **Finish** to close the wizard.

2-2-8. Click on **IP Catalog** tab in XPS and observe that **LED_IP** is added under USER sub-folder under the **Project Local PCores** repository.

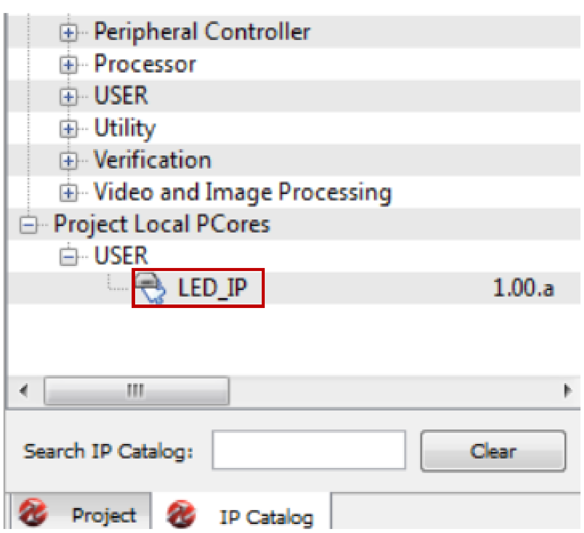

Figure 9. IP Catalog updated entry

The peripheral which you just added becomes part of the available cores list. Use Windows Explorer to browse to your project directory and ensure that the following structure has been created by the Create and Import Peripheral Wizard.

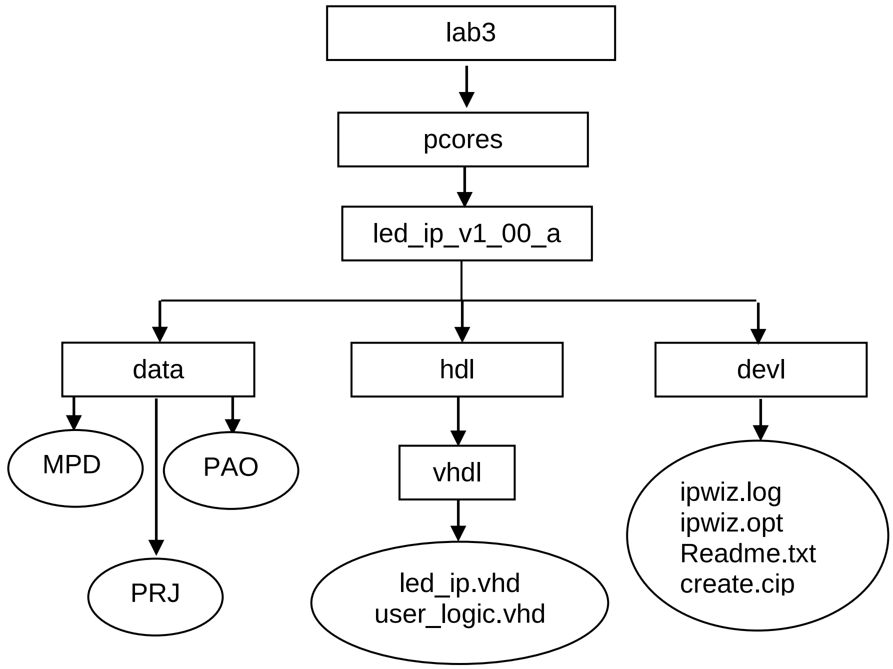

Figure 10. Structure created by the Create and Import Peripheral Wizard

Create the Peripheral — Step 3

3-1. Update the MPD file to include the led data output of the LED controller peripheral so the port can be connected in XPS.

Add a port called “led” to the MPD file.

3-1-1. Open **led_ip_v2_1_0.mpd** in the **pcores\led_ip_v1_00_a\data** under the **lab3** directory.

3-1-2. Add following line before the **S_AXI_ACLK** port under the **Ports** section.

PORT LED="", DIR=O, VEC=[7:0]

```
## Ports
PORT LED = "", DIR = O, VEC=[7:0]
PORT S_AXI_ACLK = "", DIR = I, SIGIS = CLK, BUS = S_AXI
PORT S_AXI_ARESETN = ARESETN, DIR = I, SIGIS = RST, BUS =
PORT S_AXI_AWADDR = AWADDR, DIR = I, VEC = [(C_S_AXI_ADDR
PORT S_AXI_AWVALID = AWVALID, DIR = I, BUS = S_AXI
PORT S_AXI_WDATA = WDATA, DIR = I, VEC = [(C_S_AXI_DATA_W
PORT S_AXI_WSTRB = WSTRB, DIR = I, VEC = [((C_S_AXI_DATA_
PORT S_AXI_WVALID = WVALID, DIR = I, BUS = S_AXI
```

Figure 11. Update the MPD file for the LED Controller Peripheral

3-1-3. Save the file and close it.

3-2. Create the LED controller using the appropriate HDL template files generated from the Create/Import peripheral wizard: led_ip.vhd and user_logic.vhd. You can edit these files using a standard text editor.

3-2-1. Open **led_ip.vhd** in the **pcores\ led_ip_v1_00_a\hdl\vhdl** directory.

3-2-2. Add user port **LED** of width 8 under **USER ports added here** token by search for **--USER ports added here.**

```
port
(
  -- ADD USER PORTS BELOW THIS LINE ------------------
  --USER ports added here
  LED                              : out std_logic_vector(7 downto 0);
  -- ADD USER PORTS ABOVE THIS LINE ------------------

  -- DO NOT EDIT BELOW THIS LINE ---------------------
  -- Bus protocol ports, do not add to or delete
  S_AXI_ACLK                       : in  std_logic;
  S_AXI_ARESETN                    : in  std_logic;
```

Figure 12. Add the user port LED

3-2-3. Search for next **--USER** and add port mapping statement, save the file and then close it.

```
      C_NUM_REG                      => USER_NUM_REG,
      C_SLV_DWIDTH                   => USER_SLV_DWIDTH
    )
    port map
    (
      -- MAP USER PORTS BELOW THIS LINE ------------------
      --USER ports mapped here
      LED                            => LED,
      -- MAP USER PORTS ABOVE THIS LINE ------------------

      Bus2IP_Clk                     => ipif_Bus2IP_Clk,
      Bus2IP_Resetn                  => ipif_Bus2IP_Resetn,
      Bus2IP_Data                    => ipif_Bus2IP_Data,
      Bus2IP_BE                      => ipif_Bus2IP_BE,
```

Figure 13. Add port mapping statement

3-2-4. Open **user_logic.vhd** file from the *vhdl* directory and add **LED** port definition in the USER Ports area.

```vhdl
    C_NUM_REG                      : integer              := 1;
    C_SLV_DWIDTH                   : integer              := 32
    -- DO NOT EDIT ABOVE THIS LINE ---------------------
  );
  port
  (
    -- ADD USER PORTS BELOW THIS LINE ------------------
    --USER ports added here
    LED                            : out std_logic_vector(7 downto 0);
    -- 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_Resetn                  : in  std_logic;
```

Figure 14. Add the LED port definition

3-2-5. Search for next **--USER** and the enter the internal signal declaration according to the figure below.

```vhdl

architecture IMP of user_logic is

  --USER signal declarations added here, as needed for user logic
  signal LED_i                          : std_logic_vector(7 downto 0);

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

Figure 15. Internal signal declaration for the User Logic

3-2-6. Search for **--USER logic implementation** and add the following code or copy it from lab3_user_logic.vhd file located at **c:\xup\embedded\sources** directory.

```vhdl
begin

  --USER logic implementation added here

  led_PROC : process (Bus2IP_Clk) is
  begin
    if Bus2IP_Clk'event and Bus2IP_Clk = '1' then
      if Bus2IP_Resetn = '0' then
        led_i <= (others => '0');
      else
        if Bus2IP_WrCE(0) = '1' then
          led_i <= Bus2IP_Data(7 downto 0);
        end if;
      end if;
    end if;
  end process led_PROC;
  LED <= led_i;

```

Figure 16. Add code

3-2-7. Save changes and close the **user_logic.vhd**

3-2-8. Select **Project** > **Rescan User Repositories** to have the changes in effect.

Add and Connect the Peripheral — Step 4

4-1. Add the LED to the design and connect to the AXILite bus in the System Assembly View. Make internal and external port connections. Assign an address range to it. Establish the LED data port as external FPGA pins.

4-1-1. Add the **led_ip** core to the system by double-clicking its entry in the IP Catalog.

4-1-2. Click **Yes** to add the IP instance, and **OK** twice to connect to the processor system using the default settings

4-1-3. Select the **Bus Interfaces** tab in the System Assembly View and verify that the peripheral is connected to AXIlite_0.

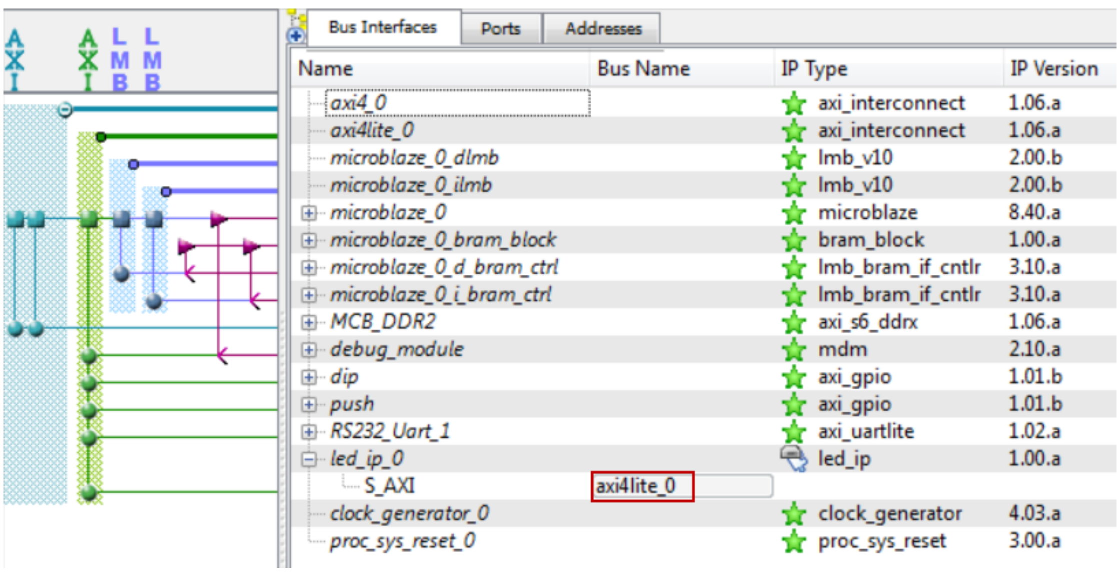

Figure 17. Making bus connection

4-1-4. Select the **Ports** tab, right-click the **led** port of the **led_ip_0** instance and select **Make External**.

4-1-5. Select **Addresses** filter and verify that the address is assigned to the led_ip instance.

The Addresses tab should look similar to as shown.

Bus Interfaces | Ports | Addresses

Instance	Base Name	Base Address	High Address	Size
microblaze_0's Address Map				
microblaze_0_d_bram_ctrl	C_BASEADDR	0x00000000	0x00001FFF	8K
microblaze_0_i_bram_ctrl	C_BASEADDR	0x00000000	0x00001FFF	8K
dip	C_BASEADDR	0x40000000	0x4000FFFF	64K
push	C_BASEADDR	0x40040000	0x4004FFFF	64K
RS232_Uart_1	C_BASEADDR	0x40600000	0x4060FFFF	64K
debug_module	C_BASEADDR	0x41400000	0x4140FFFF	64K
led_ip_0	C_BASEADDR	0x7F400000	0x7F40FFFF	64K
MCB_DDR2	C_S0_AXI_BASE...	0xA8000000	0xAFFFFFFF	128M

Figure 18. Generate Addresses

4-2. Modify the system.ucf file to assign external LED controller connections to the proper FPGA pin locations.

4-2-1. Open the **system.ucf** file by double-clicking the **UCF File: data\system.ucf** entry under Project Files in the Project tab.

4-2-2. Open the ***C:\xup\embedded\sources\lab3.ucf*** file and copy the pin assignments under the original ucf file.

```
#----------------------------------------------------------------------
# IO Pad Location Constraints / Properties for Character LED GPIO
#----------------------------------------------------------------------

NET led_ip_0_LED_pin<0> LOC = U18 | IOSTANDARD = LVCMOS33 ; # LED0
NET led_ip_0_LED_pin<1> LOC = M14 | IOSTANDARD = LVCMOS33 ; # LED1
NET led_ip_0_LED_pin<2> LOC = N14 | IOSTANDARD = LVCMOS33 ; # LED2
NET led_ip_0_LED_pin<3> LOC = L14 | IOSTANDARD = LVCMOS33 ; # LED3
NET led_ip_0_LED_pin<4> LOC = M13 | IOSTANDARD = LVCMOS33 ; # LED4
NET led_ip_0_LED_pin<5> LOC = D4  | IOSTANDARD = LVCMOS33 ; # LED5
NET led_ip_0_LED_pin<6> LOC = P16 | IOSTANDARD = LVCMOS33 ; # LED6
NET led_ip_0_LED_pin<7> LOC = N12 | IOSTANDARD = LVCMOS33 ; # LED7
```

Figure 19. Adding UCF constraints

4-2-3. Save and close the file

4-2-4. Click on **Hardware > Generate Bitstream.**

4-2-5. Close the project after Bitstream generation is done.

We will do the software development in Lab 4 and verify the functionality in hardware.

Conclusion

The Create and Import Peripheral Wizard was used to create peripheral templates for the AXILite interface. Logic was added to the templates to create an LED interface peripheral. The peripheral was then integrated into an existing processor system.

Completed MHS File

```
# ##############################################################################
# Created by Base System Builder Wizard for Xilinx EDK 14.2 Build EDK_P.28xd
# Wed Jul 25 08:27:29 2012
# Target Board:  digilent atlys Rev C
# Family:    spartan6
# Device:    xc6slx45
# Package:   csg324
# Speed Grade:  -2
# ##############################################################################
 PARAMETER VERSION = 2.1.0


 PORT zio = zio, DIR = IO
 PORT rzq = rzq, DIR = IO
 PORT mcbx_dram_we_n = mcbx_dram_we_n, DIR = O
 PORT mcbx_dram_udqs_n = mcbx_dram_udqs_n, DIR = IO
 PORT mcbx_dram_udqs = mcbx_dram_udqs, DIR = IO
 PORT mcbx_dram_udm = mcbx_dram_udm, DIR = O
 PORT mcbx_dram_ras_n = mcbx_dram_ras_n, DIR = O
 PORT mcbx_dram_odt = mcbx_dram_odt, DIR = O
 PORT mcbx_dram_ldm = mcbx_dram_ldm, DIR = O
 PORT mcbx_dram_dqs_n = mcbx_dram_dqs_n, DIR = IO
 PORT mcbx_dram_dqs = mcbx_dram_dqs, DIR = IO
 PORT mcbx_dram_dq = mcbx_dram_dq, DIR = IO, VEC = [15:0]
 PORT mcbx_dram_clk_n = mcbx_dram_clk_n, DIR = O, SIGIS = CLK
 PORT mcbx_dram_clk = mcbx_dram_clk, DIR = O, SIGIS = CLK
 PORT mcbx_dram_cke = mcbx_dram_cke, DIR = O
 PORT mcbx_dram_cas_n = mcbx_dram_cas_n, DIR = O
 PORT mcbx_dram_ba = mcbx_dram_ba, DIR = O, VEC = [2:0]
 PORT mcbx_dram_addr = mcbx_dram_addr, DIR = O, VEC = [12:0]
 PORT RS232_Uart_1_sout = RS232_Uart_1_sout, DIR = O
 PORT RS232_Uart_1_sin = RS232_Uart_1_sin, DIR = I
 PORT RESET = RESET, DIR = I, SIGIS = RST, RST_POLARITY = 0
 PORT GCLK = GCLK, DIR = I, SIGIS = CLK, CLK_FREQ = 100000000
 PORT dip_GPIO_IO_I_pin = dip_GPIO_IO_I, DIR = I, VEC = [7:0]
 PORT push_GPIO_IO_I_pin = push_GPIO_IO_I, DIR = I, VEC = [4:0]
 PORT led_ip_0_LED_pin = led_ip_0_LED, DIR = O, VEC = [7:0]


BEGIN proc_sys_reset
 PARAMETER INSTANCE = proc_sys_reset_0
 PARAMETER HW_VER = 3.00.a
 PARAMETER C_EXT_RESET_HIGH = 0
 PORT MB_Debug_Sys_Rst = proc_sys_reset_0_MB_Debug_Sys_Rst
 PORT Dcm_locked = proc_sys_reset_0_Dcm_locked
 PORT MB_Reset = proc_sys_reset_0_MB_Reset
 PORT Slowest_sync_clk = clk_100_0000MHzPLL0
 PORT Interconnect_aresetn = proc_sys_reset_0_Interconnect_aresetn
 PORT Ext_Reset_In = RESET
 PORT BUS_STRUCT_RESET = proc_sys_reset_0_BUS_STRUCT_RESET
END

BEGIN lmb_v10
 PARAMETER INSTANCE = microblaze_0_ilmb
 PARAMETER HW_VER = 2.00.b
```

```
 PORT SYS_RST = proc_sys_reset_0_BUS_STRUCT_RESET
 PORT LMB_CLK = clk_100_0000MHzPLL0
END

BEGIN lmb_bram_if_cntlr
 PARAMETER INSTANCE = microblaze_0_i_bram_ctrl
 PARAMETER HW_VER = 3.10.a
 PARAMETER C_BASEADDR = 0x00000000
 PARAMETER C_HIGHADDR = 0x00001fff
 BUS_INTERFACE SLMB = microblaze_0_ilmb
 BUS_INTERFACE BRAM_PORT = microblaze_0_i_bram_ctrl_2_microblaze_0_bram_block
END

BEGIN lmb_v10
 PARAMETER INSTANCE = microblaze_0_dlmb
 PARAMETER HW_VER = 2.00.b
 PORT SYS_RST = proc_sys_reset_0_BUS_STRUCT_RESET
 PORT LMB_CLK = clk_100_0000MHzPLL0
END

BEGIN lmb_bram_if_cntlr
 PARAMETER INSTANCE = microblaze_0_d_bram_ctrl
 PARAMETER HW_VER = 3.10.a
 PARAMETER C_BASEADDR = 0x00000000
 PARAMETER C_HIGHADDR = 0x00001fff
 BUS_INTERFACE SLMB = microblaze_0_dlmb
 BUS_INTERFACE BRAM_PORT = microblaze_0_d_bram_ctrl_2_microblaze_0_bram_block
END

BEGIN bram_block
 PARAMETER INSTANCE = microblaze_0_bram_block
 PARAMETER HW_VER = 1.00.a
 BUS_INTERFACE PORTA = microblaze_0_i_bram_ctrl_2_microblaze_0_bram_block
 BUS_INTERFACE PORTB = microblaze_0_d_bram_ctrl_2_microblaze_0_bram_block
END

BEGIN microblaze
 PARAMETER INSTANCE = microblaze_0
 PARAMETER HW_VER = 8.40.a
 PARAMETER C_INTERCONNECT = 2
 PARAMETER C_USE_BARREL = 1
 PARAMETER C_USE_FPU = 0
 PARAMETER C_DEBUG_ENABLED = 1
 PARAMETER C_ICACHE_BASEADDR = 0xa8000000
 PARAMETER C_ICACHE_HIGHADDR = 0xafffffff
 PARAMETER C_USE_ICACHE = 1
 PARAMETER C_CACHE_BYTE_SIZE = 8192
 PARAMETER C_ICACHE_ALWAYS_USED = 1
 PARAMETER C_DCACHE_BASEADDR = 0xa8000000
 PARAMETER C_DCACHE_HIGHADDR = 0xafffffff
 PARAMETER C_USE_DCACHE = 1
 PARAMETER C_DCACHE_BYTE_SIZE = 8192
 PARAMETER C_DCACHE_ALWAYS_USED = 1
 BUS_INTERFACE ILMB = microblaze_0_ilmb
 BUS_INTERFACE DLMB = microblaze_0_dlmb
 BUS_INTERFACE M_AXI_DP = AXIlite_0
 BUS_INTERFACE M_AXI_DC = AXI_0
 BUS_INTERFACE M_AXI_IC = AXI_0
```

```
 BUS_INTERFACE DEBUG = microblaze_0_debug
 PORT MB_RESET = proc_sys_reset_0_MB_Reset
 PORT CLK = clk_100_0000MHzPLL0
END

BEGIN mdm
 PARAMETER INSTANCE = debug_module
 PARAMETER HW_VER = 2.10.a
 PARAMETER C_INTERCONNECT = 2
 PARAMETER C_USE_UART = 1
 PARAMETER C_BASEADDR = 0x41400000
 PARAMETER C_HIGHADDR = 0x4140ffff
 BUS_INTERFACE S_AXI = AXIlite_0
 BUS_INTERFACE MBDEBUG_0 = microblaze_0_debug
 PORT Debug_SYS_Rst = proc_sys_reset_0_MB_Debug_Sys_Rst
 PORT S_AXI_ACLK = clk_100_0000MHzPLL0
END

BEGIN clock_generator
 PARAMETER INSTANCE = clock_generator_0
 PARAMETER HW_VER = 4.03.a
 PARAMETER C_EXT_RESET_HIGH = 0
 PARAMETER C_CLKIN_FREQ = 100000000
 PARAMETER C_CLKOUT0_FREQ = 600000000
 PARAMETER C_CLKOUT0_GROUP = PLL0
 PARAMETER C_CLKOUT0_BUF = FALSE
 PARAMETER C_CLKOUT1_FREQ = 600000000
 PARAMETER C_CLKOUT1_PHASE = 180
 PARAMETER C_CLKOUT1_GROUP = PLL0
 PARAMETER C_CLKOUT1_BUF = FALSE
 PARAMETER C_CLKOUT2_FREQ = 100000000
 PARAMETER C_CLKOUT2_GROUP = PLL0
 PORT LOCKED = proc_sys_reset_0_Dcm_locked
 PORT CLKOUT2 = clk_100_0000MHzPLL0
 PORT RST = RESET
 PORT CLKOUT0 = clk_600_0000MHzPLL0_nobuf
 PORT CLKOUT1 = clk_600_0000MHz180PLL0_nobuf
 PORT CLKIN = GCLK
END

BEGIN axi_interconnect
 PARAMETER INSTANCE = AXIlite_0
 PARAMETER HW_VER = 1.06.a
 PARAMETER C_INTERCONNECT_CONNECTIVITY_MODE = 0
 PORT INTERCONNECT_ARESETN = proc_sys_reset_0_Interconnect_aresetn
 PORT INTERCONNECT_ACLK = clk_100_0000MHzPLL0
END

BEGIN axi_interconnect
 PARAMETER INSTANCE = AXI_0
 PARAMETER HW_VER = 1.06.a
 PORT interconnect_aclk = clk_100_0000MHzPLL0
 PORT INTERCONNECT_ARESETN = proc_sys_reset_0_Interconnect_aresetn
END

BEGIN axi_uartlite
 PARAMETER INSTANCE = RS232_Uart_1
 PARAMETER HW_VER = 1.02.a
```

```
 PARAMETER C_BAUDRATE = 115200
 PARAMETER C_DATA_BITS = 8
 PARAMETER C_USE_PARITY = 0
 PARAMETER C_ODD_PARITY = 1
 PARAMETER C_BASEADDR = 0x40600000
 PARAMETER C_HIGHADDR = 0x4060ffff
 BUS_INTERFACE S_AXI = AXIlite_0
 PORT S_AXI_ACLK = clk_100_0000MHzPLL0
 PORT TX = RS232_Uart_1_sout
 PORT RX = RS232_Uart_1_sin
END

BEGIN axi_s6_ddrx
 PARAMETER INSTANCE = MCB_DDR2
 PARAMETER HW_VER = 1.06.a
 PARAMETER C_MCB_RZQ_LOC = L6
 PARAMETER C_MCB_ZIO_LOC = C2
 PARAMETER C_MEM_TYPE = DDR2
 PARAMETER C_MEM_PARTNO = EDE1116AXXX-8E
 PARAMETER C_MEM_BANKADDR_WIDTH = 3
 PARAMETER C_MEM_NUM_COL_BITS = 10
 PARAMETER C_SKIP_IN_TERM_CAL = 0
 PARAMETER C_S0_AXI_ENABLE = 1
 PARAMETER C_INTERCONNECT_S0_AXI_MASTERS = microblaze_0.M_AXI_DC &
microblaze_0.M_AXI_IC
 PARAMETER C_MEM_DDR2_RTT = 50OHMS
 PARAMETER C_S0_AXI_STRICT_COHERENCY = 0
 PARAMETER C_INTERCONNECT_S0_AXI_AW_REGISTER = 8
 PARAMETER C_INTERCONNECT_S0_AXI_AR_REGISTER = 8
 PARAMETER C_INTERCONNECT_S0_AXI_W_REGISTER = 8
 PARAMETER C_INTERCONNECT_S0_AXI_R_REGISTER = 8
 PARAMETER C_INTERCONNECT_S0_AXI_B_REGISTER = 8
 PARAMETER C_S0_AXI_BASEADDR = 0xa8000000
 PARAMETER C_S0_AXI_HIGHADDR = 0xafffffff
 BUS_INTERFACE S0_AXI = AXI_0
 PORT zio = zio
 PORT rzq = rzq
 PORT s0_axi_aclk = clk_100_0000MHzPLL0
 PORT ui_clk = clk_100_0000MHzPLL0
 PORT mcbx_dram_we_n = mcbx_dram_we_n
 PORT mcbx_dram_udqs_n = mcbx_dram_udqs_n
 PORT mcbx_dram_udqs = mcbx_dram_udqs
 PORT mcbx_dram_udm = mcbx_dram_udm
 PORT mcbx_dram_ras_n = mcbx_dram_ras_n
 PORT mcbx_dram_odt = mcbx_dram_odt
 PORT mcbx_dram_ldm = mcbx_dram_ldm
 PORT mcbx_dram_dqs_n = mcbx_dram_dqs_n
 PORT mcbx_dram_dqs = mcbx_dram_dqs
 PORT mcbx_dram_dq = mcbx_dram_dq
 PORT mcbx_dram_clk_n = mcbx_dram_clk_n
 PORT mcbx_dram_clk = mcbx_dram_clk
 PORT mcbx_dram_cke = mcbx_dram_cke
 PORT mcbx_dram_cas_n = mcbx_dram_cas_n
 PORT mcbx_dram_ba = mcbx_dram_ba
 PORT mcbx_dram_addr = mcbx_dram_addr
 PORT sysclk_2x = clk_600_0000MHzPLL0_nobuf
 PORT sysclk_2x_180 = clk_600_0000MHz180PLL0_nobuf
 PORT SYS_RST = proc_sys_reset_0_BUS_STRUCT_RESET
```

```
 PORT PLL_LOCK = proc_sys_reset_0_Dcm_locked
END

BEGIN axi_gpio
 PARAMETER INSTANCE = dip
 PARAMETER HW_VER = 1.01.b
 PARAMETER C_GPIO_WIDTH = 8
 PARAMETER C_ALL_INPUTS = 1
 PARAMETER C_BASEADDR = 0x40000000
 PARAMETER C_HIGHADDR = 0x4000ffff
 BUS_INTERFACE S_AXI = AXIlite_0
 PORT S_AXI_ACLK = clk_100_0000MHzPLL0
 PORT GPIO_IO_I = dip_GPIO_IO_I
END

BEGIN axi_gpio
 PARAMETER INSTANCE = push
 PARAMETER HW_VER = 1.01.b
 PARAMETER C_GPIO_WIDTH = 5
 PARAMETER C_ALL_INPUTS = 1
 PARAMETER C_BASEADDR = 0x40040000
 PARAMETER C_HIGHADDR = 0x4004ffff
 BUS_INTERFACE S_AXI = AXIlite_0
 PORT S_AXI_ACLK = clk_100_0000MHzPLL0
 PORT GPIO_IO_I = push_GPIO_IO_I
END

BEGIN led_ip
 PARAMETER INSTANCE = led_ip_0
 PARAMETER HW_VER = 1.00.a
 PARAMETER C_BASEADDR = 0x7f400000
 PARAMETER C_HIGHADDR = 0x7f40ffff
 BUS_INTERFACE S_AXI = AXIlite_0
 PORT S_AXI_ACLK = clk_100_0000MHzPLL0
 PORT LED = led_ip_0_LED
END
```