

# Custom Peripheral for the AXI4-Lite Interface

## OBJECTIVES

- Create a custom VHDL peripheral with an AXI4-Lite Interface.
- Integrate the VHDL peripheral in a Block Based Design in Vivado.
- Create a software application in SDK that can handle the custom peripheral.

## ZYBO BOARD SETUP FOR HARDWARE/SOFTWARE CO-DESIGN

- The current Zynq Book Tutorials (Aug 15<sup>th</sup>, 2015) includes a procedure that requires copying definition files into the Vivado installation directory. This helps when setting up the ZYBO Board.
- In this tutorial, we will manually indicate the Zynq device and the Processing System (PS) definition file.
- ZYBO Board: PS\_CLK input: 50 MHz (by default it generates a PL clock of 100 MHz). External PL\_CLK input: 125 MHz.
- The procedure described here resembles that of the Zynq Book Tutorials ([www.zynqbook.com](http://www.zynqbook.com)). Some modifications were made due to the use of the ZYBO Board.

## CUSTOM PERIPHERAL AND BLOCK DESIGN PROJECT IN VIVADO

- Refer to the Zynq Book Tutorial: IP → Creating IP in VHDL for detailed step-by-step instructions.

## PIXEL PROCESSOR: CUSTOM PERIPHERAL FOR AXI4-LITE

- We will use the Pixel Processor with  $NC = 4, NI = NO = 8$ . Files to use:
  - ✓ `static_ip.vhd`: top file for the Pixel Processor IP.
  - ✓ `LUT_group.vhd`, `LUT_NItO.vhd`, `LUT_NItO1.vhd`, `pack_xtras.vhd`: Files that make up the Pixel Processor.
  - ✓ `LUT_values8to8.txt`: LUT values.
- We need two Slave Registers to process data through this Pixel Processor circuit (one for writing data, one for reading data).
- Create a new project in Vivado. Select the **ZYNQ XC7Z010-1CLG400** device.
- Select Default Language: VHDL. This way, the system wrapper and the template files for the AXI4-Lite peripherals are created in VHDL.
- From the menu bar, select **Tools** → **Create and Package IP**. A new Vivado project will open.
  - ✓ Create a new AXI4 Peripheral. Name: `mypix`. Location `/ip_repo`.
  - Peripheral Repositories tip:* To add a previously-generated IP into a new project, go to: Project Settings → IP → IP repositories and point to the repository folder. This is important if that repository was created when working on a different project.
  - ✓ Add Interface: Lite, 32 bits, 4 registers (we just need 2, but 4 is the minimum).
  - ✓ Select Edit IP. A New project appears, open it and look for the `<peripheral_name>_S00_AXI.vhd` file (in this case it will be `my_pix_v1_0_S00_AXI.vhd`). Modify that file by i) including the pixel processor (add new files to the folder `/hdl` in `/mypix_1.0`) in the VHDL code, and ii) using only the required registers, i.e., commenting out VHDL code that specifies unused registers. As a shortcut, you can just use the `my_pix_v1_0_S00_AXI.vhd` file that is available for download.
  - ✓ There is no need to add ports as our peripheral does not include external I/Os.
  - ✓ Follow instructions on the Zynq Book Tutorial (Return to IP Packager, Review and Package).
- You will return to the original Vivado Project.

## PIXEL PROCESSOR: CREATING A BLOCK DESIGN PROJECT IN VIVADO

- Click on 'Create Block Design' and instantiate the Zynq PS and the AXI MYPPIX peripherals.
- Click on 'Run Block Automation' and 'Run Connection Automation'.
- Double click on ZYNQ PS: Load the `ZYBO_zynq_def.xml` file (Import XPS Settings). This indicates which peripherals are used by the PS. If the file is not loaded, the software application on the ddr or a peripheral (e.g. UART) will not work properly.
- We do not need to add the `.xdc` file as our peripheral does not use external ports.
- Create the VHDL wrapper (Sources Window → right click on the top-level system design).
- Synthesize, implement, and generate the bitstream.
  - ✓ It will not work at first. But the `/ipshared` folder will be created in the project folder. We need to place any ancillary files (e.g. `.txt` files) in the `/hdl` portion of that folder to make it work.
- Export hardware (with bitstream) and launch SDK

## PIXEL PROCESSOR: SOFTWARE APPLICATION IN SDK

- Use Tutorial Unit 2 for instructions on how to create and test a software application on SDK.
- Navigate to Xilinx Tools → Repositories, click on 'New' and then browse to the folder `/ip_repo/mypix_1.0` and click ok.
- Create a new SDK application. Then, copy the following file into the `/src` folder: `pixproc_test.c`. This file will test all the possible inputs to each 8-bit LUT (0x00 to 0xFF): The 32-bit input word will have four identical bytes.

## PIPELINED DIVIDER: CUSTOM PERIPHERAL FOR AXI4-LITE

- We will use the Pipelined Integer Divider with  $N = 16, M = 16$ . Files to use:
  - ✓ `mydiv_v1_0.vhd`: AXI4-Lite Peripheral (top file). This is the same file generated by Vivado.
  - ✓ `mydiv_v1_0_s00_AXI.vhd`: AXI4-Lite Interface description. This file is generated by Vivado, but edited to include the Pipelined Divider IP.
  - ✓ `divpip_ip.vhd`: Pipelined Divider with some AXI4-Lite Interfacing (FSM).
  - ✓ `res_div_pip.vhd`: top file of the Pipelined Divider IP.
  - ✓ `full_add.vhd`, `my_pashiftreg.vhd`, `unit_proc.vhd`, `dfte.vhd`: Files that make up the Pipelined Divider.
- We need 3 Slave Registers to process data through this Pipelined Divider circuit (one for writing data, two for reading data).
- Create a new project in Vivado. Select the **ZYNQ XC7Z010-1CLG400** device.
- Select Default Language: VHDL. This way, the system wrapper and the template files for the AXI4-Lite peripherals are created in VHDL.
- From the menu bar, select **Tools** → **Create and Package IP**. A new Vivado project will open.
  - ✓ Create a new AXI4 Peripheral. Name: `mydiv`. Location `/ip_repo`.  
*Peripheral Repositories tip:* To add a previously-generated IP into a new project, go to: Project Settings → IP → IP repositories and point to the repository folder. This is important if that repository was created when working on a different project.
  - ✓ Add Interface: Lite, 32 bits, 4 registers (we just need 2, but 4 is the minimum).
  - ✓ Select Edit IP. A New project appears, open it and look for the `<peripheral name>_S00_AXI.vhd` file (in this case it will be `my_div_v1_0_S00_AXI.vhd`). Modify that file by i) including the pipelined divider (add new files to the folder `/hdl` in `/mydiv_1.0`) in the VHDL code, and ii) using only the required registers, i.e., commenting out VHDL code that specifies unused registers. As a shortcut, you can just use the `my_div_v1_0_S00_AXI.vhd` file that is available for download.
  - ✓ There is no need to add ports as our peripheral does not include external I/Os.
  - ✓ Follow instructions on the Zynq Book Tutorial (Return to IP Packager, Review and Package).
- You will return to the original Vivado Project.

## PIPELINED DIVIDER: CREATING A BLOCK DESIGN PROJECT IN VIVADO

- Click on 'Create Block Design' and instantiate the Zynq PS and the AXI MYDIV peripherals.
- Click on 'Run Block Automation' and 'Run Connection Automation'.
- Double click on ZYNQ PS: Load the `ZYBO_zynq_def.xml` file (Import XPS Settings). This indicates which peripherals are used by the PS. If the file is not loaded, the software application on the ddr or a peripheral (e.g. UART) will not work properly.
- We do not need to add the `.xdc` file as our peripheral does not use external ports.
- Create the VHDL wrapper (Sources Window → right click on the top-level system design).
- Synthesize, implement, and generate the bitstream.
- Export hardware (with bitstream) and launch SDK

## PIPELINED DIVIDER: SOFTWARE APPLICATION IN SDK

- Use Tutorial Unit 2 for instructions on how to create and test a software application on SDK.
- Navigate to Xilinx Tools → Repositories, click on 'New' and then browse to the folder `\ip_repo\mydiv_1.0` and click ok.
- Create a new SDK application. Then, copy the following file into the `/src` folder: `div_test.c`. This file will test three integer divisions:
  - ✓ `A = 0x008C, B = 0x0009`. Expected Result: `Q = 0x000F, R = 0x0005`.
  - ✓ `A = 0x00BB, B = 0x000A`. Expected Result: `Q = 0x0012, R = 0x0007`.
  - ✓ `A = 0x0FEA, B = 0x0371`. Expected Result: `Q = 0x0004, R = 0x0226`.