PS and PL communication using AXI DMA

For the course 1DT109 Accelerating System with Programmable Logic Components, 2022.
Department IT, Uppsala University

Contact: yuan.yao@it.uu.se; shiming.li@it.uu.se

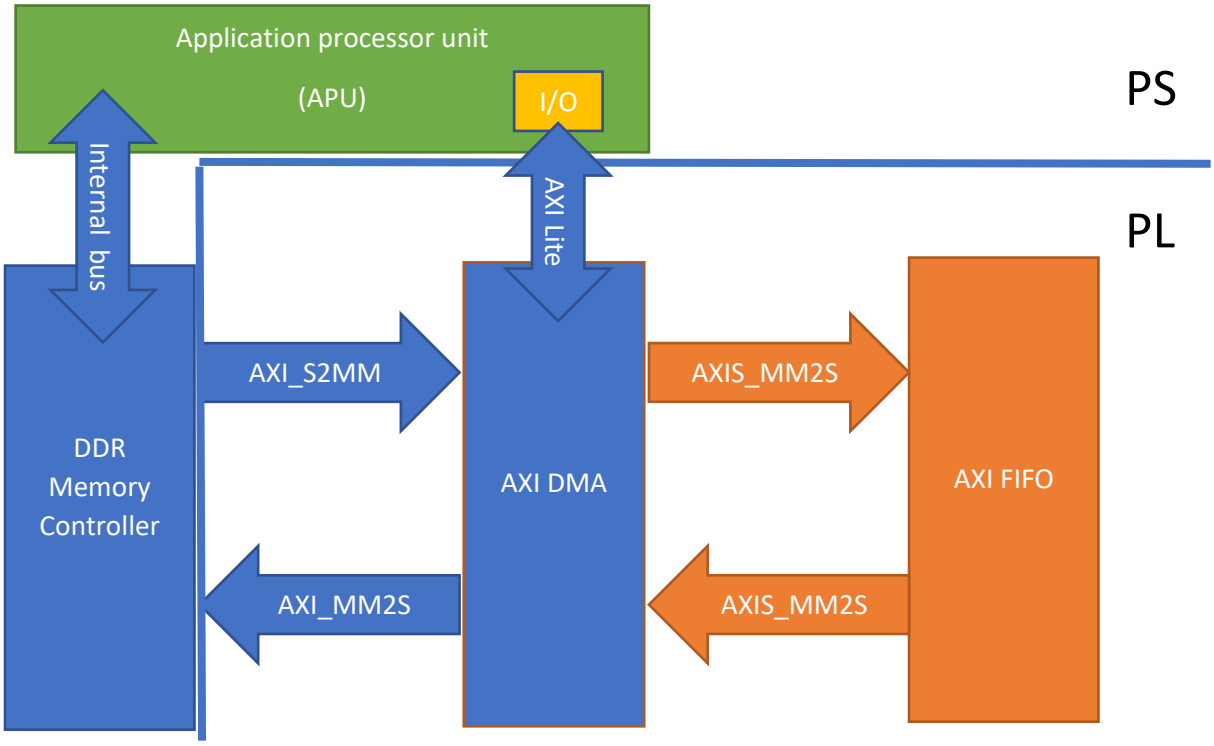
This document will walk you through

1. Understand what DMA is and why it is important for FPGA
2. How to transfer data between PS and PL using DMA with the correct components from the FPGA
3. How to exploit the above DMA feature in software (using the C programming language)

Overview: What is DMA and why we should use it for FPGA

In this design, we'll use the DMA to transfer data from memory to an IP block and back to the memory. In principle, the IP block could be any kind of data producer/consumer such as the HDR-NN hardware implement or just a matrix multiplier, but in this tutorial, we will use a simple FIFO to create a loopback. After, you'll be able to break the loop and insert whatever custom IP you like.

//TODO: think about what you want to implement in the PL side. You can implement the whole HDR-NN using Verilog and invoke it from the C. Or you can implement your HDR-NN in software and ship part of the functionalities to PL.



The block diagram above illustrates the design that we'll create. The processor and DDR memory controller are contained within the Zynq PS. The AXI DMA and AXI Data FIFO are implemented in the Zynq PL. The **AXI-lite** bus allows the processor to communicate with the AXI DMA to setup, initiate and monitor data transfers. The **AXI_MM2S** and **AXI_S2MM** are memory mapped AXI4 buses and provide the DMA access to the DDR memory. The **AXIS_MM2S** and **AXIS_S2MM** are AXI4-streaming buses, which source and sink a continuous stream of data, without addresses.

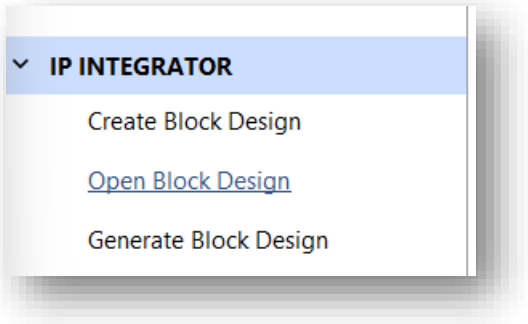
Notes:

- MM2S stands for Memory-Mapped to Streaming, whereas S2MM stands for Streaming to Memory-Mapped.
- When Scatter-Gather is used, there is an extra AXI bus between the DMA and the memory controller. It was left out of the diagram for simplicity.
- We'll start this tutorial with the base system project for the MiniZed that you've created in project lecture 2.

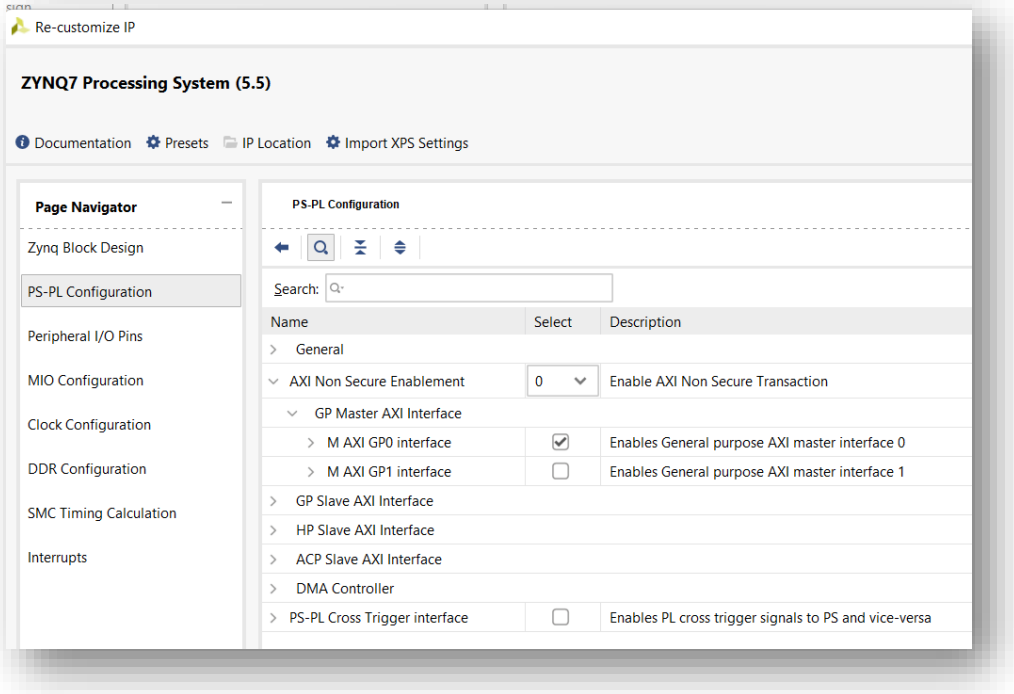
PART 1 – Setting up the DMA communication hardware platform

1, Add the AXI DMA

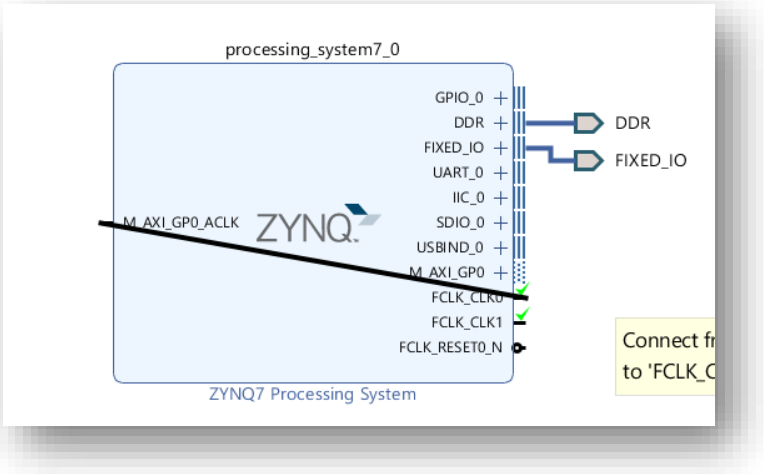
1. Open the base project in Vivado.
2. In the Flow Nagigator, click "Open Block Design".



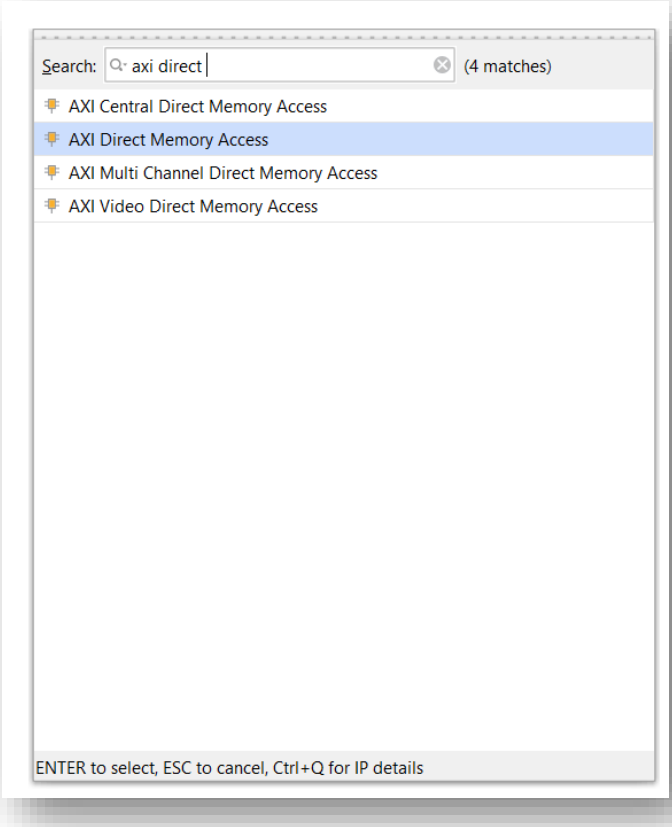
3. The block diagram should open, and you should only have the Zynq PS in the design.
4. Make sure that the Zynq PS has master GP0 enabled.



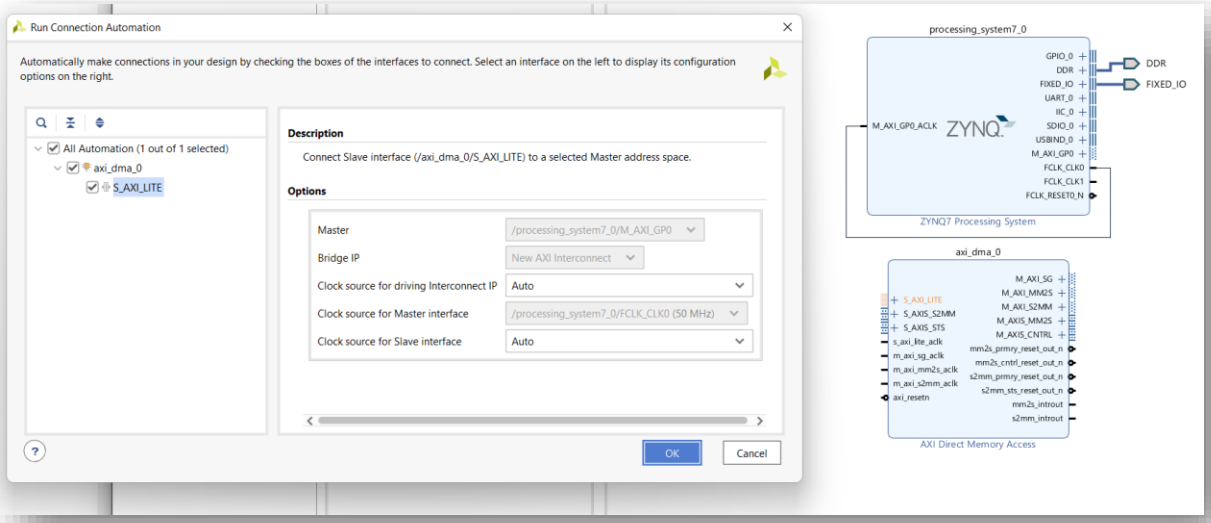
5. Connect the M_AXI_GP0_ACLK to the FCLK_CLK0 pin, which will be the driving clock of the master GP0 interface.



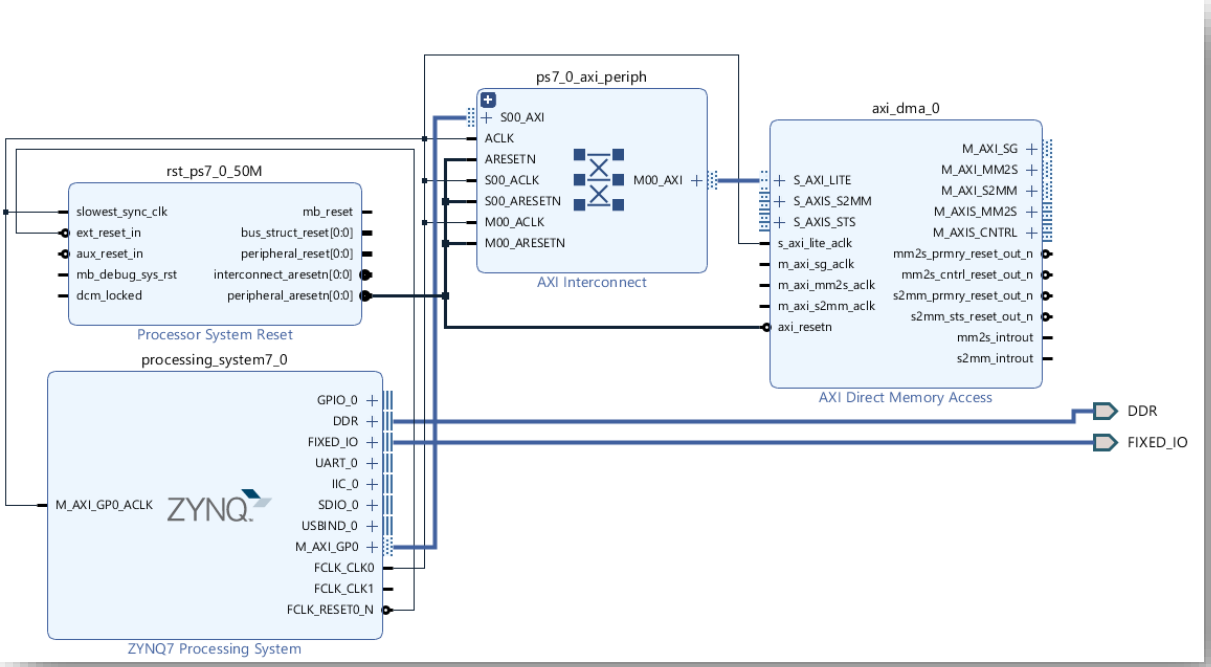
6. Click the “Add IP” icon and double click “AXI Direct Memory Access” from the catalog.



7. The DMA block should appear and designer assistance should be available. Click the “Run Connection Automation” link and **select /axi_dma_0/S_AXI_LITE** from the drop-down menu.

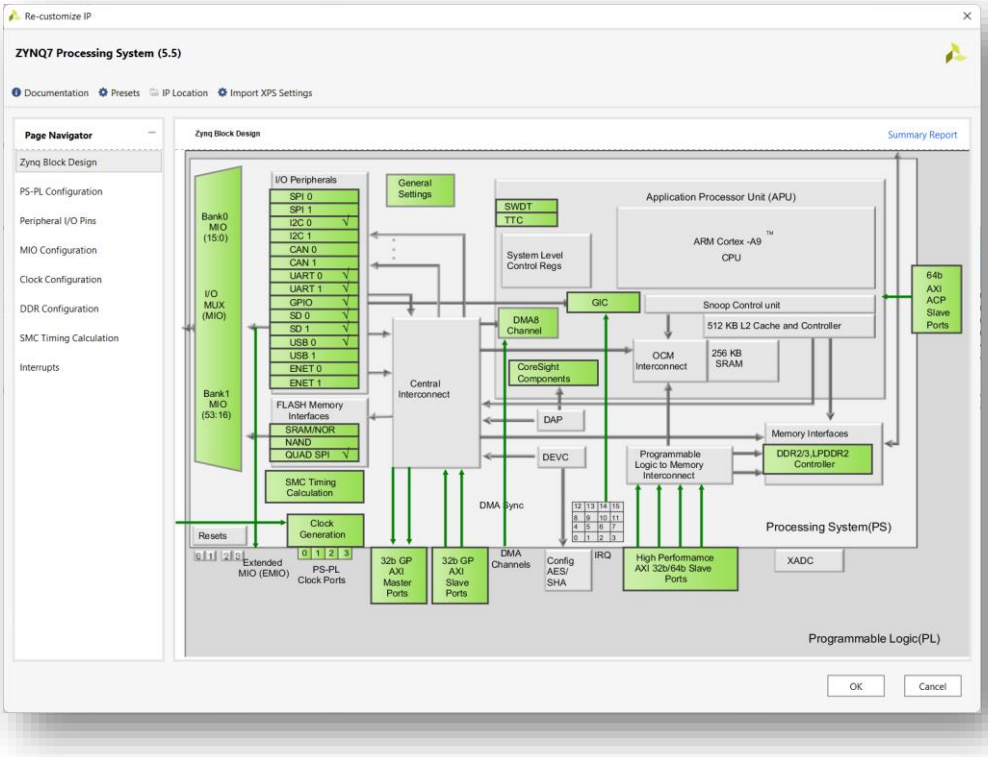


8. Click “OK” in the window that appears. Vivado will connect the AXI-lite bus of the DMA to the General Purpose AXI Interconnect of the PS. Your block diagram should now look like the following.

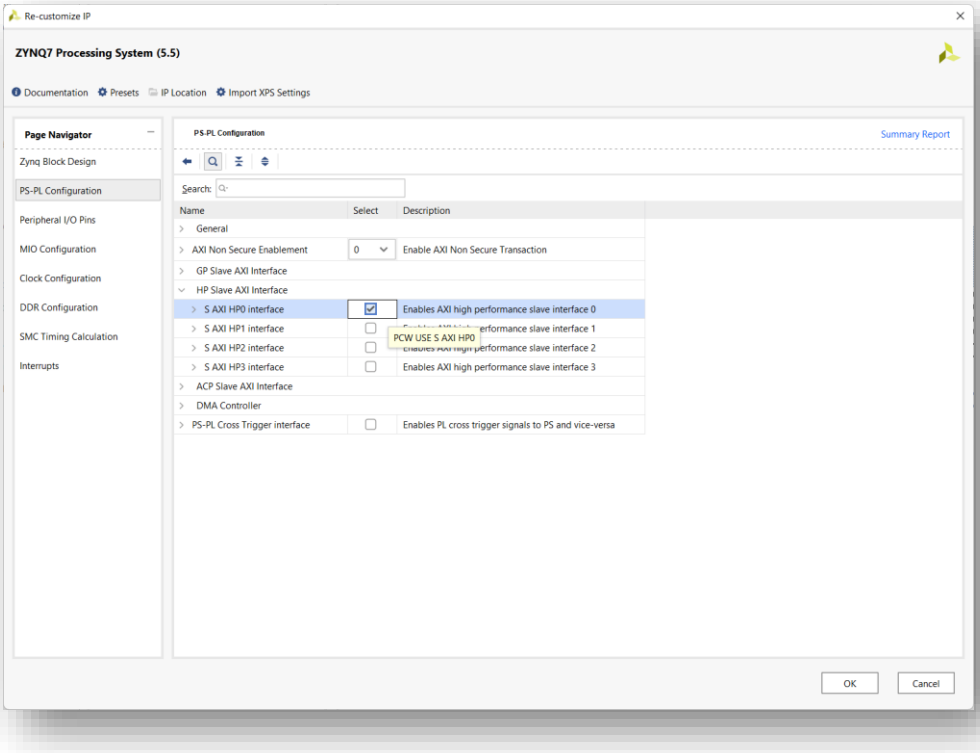


2, Connect memory controller to the DMA

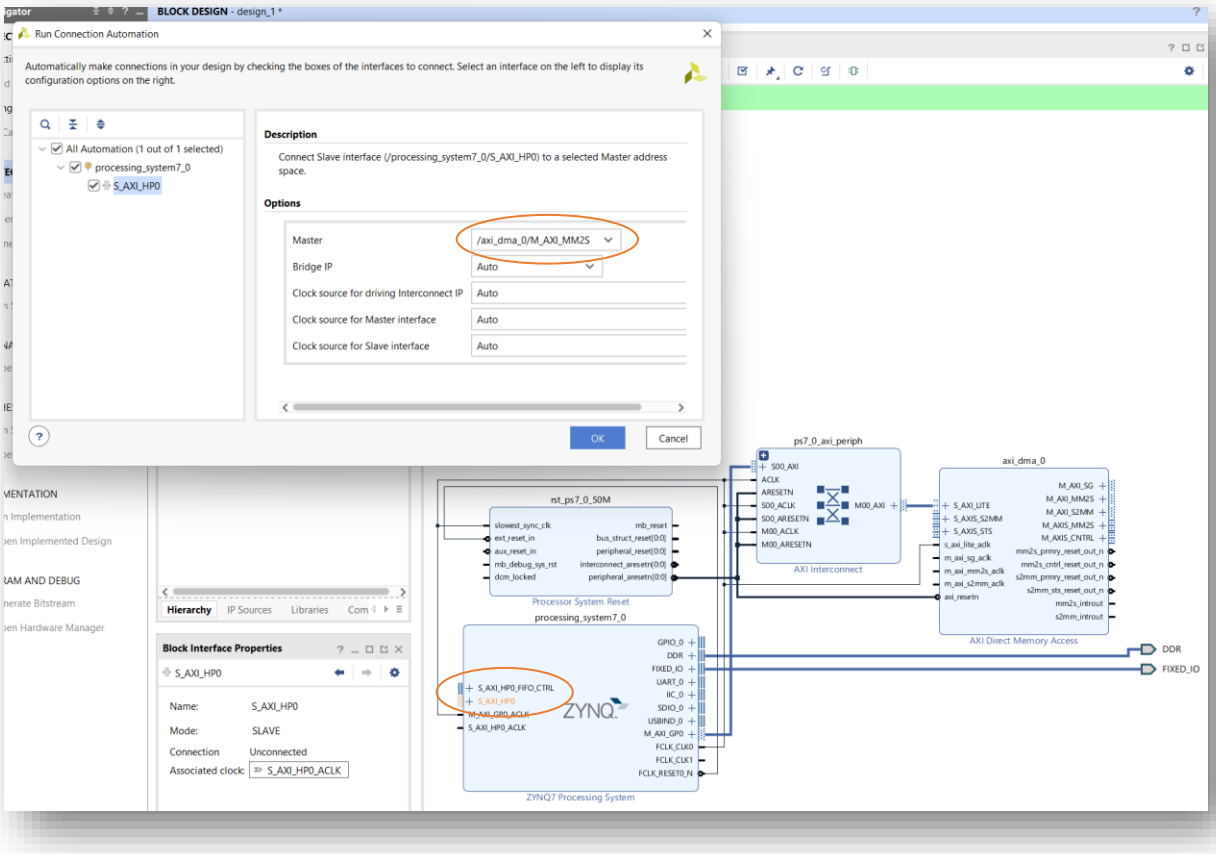
1. Now we need to connect AXI buses **M_AXI_SG**, **M_AXI_MM2S** and **M_AXI_S2MM** of the DMA to a high performance AXI slave interface on the PS. Our PS doesn’t seem to have a high-performance AXI slave interface, so we need to change the Zynq configuration to enable one. Double click on the Zynq block.



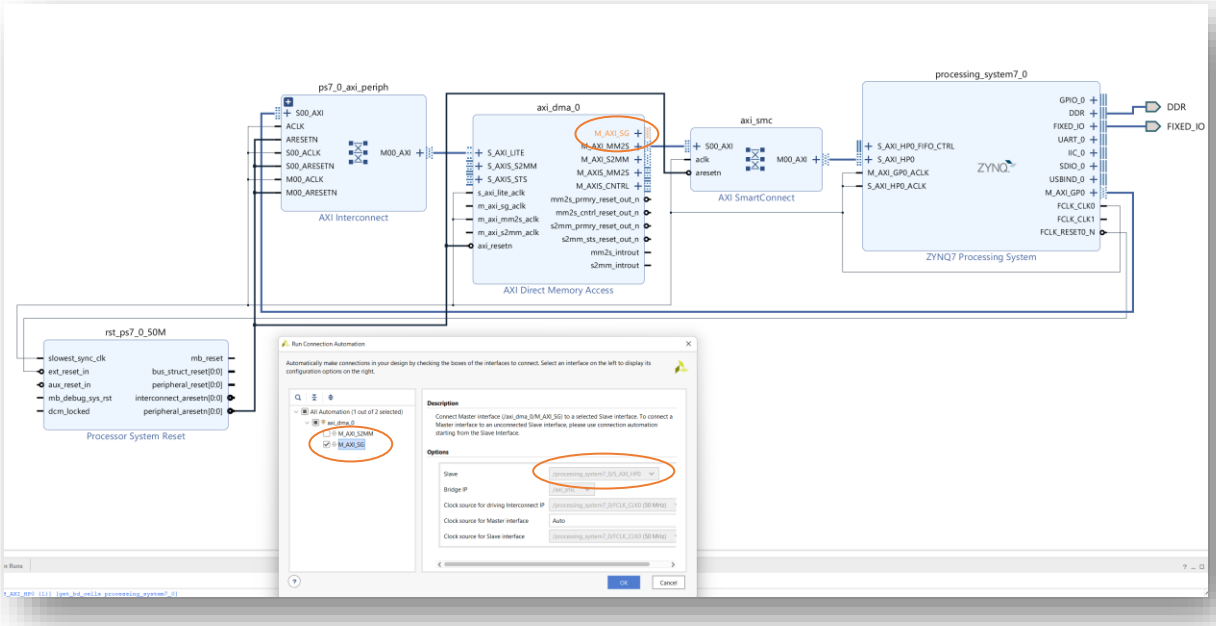
2. Select “PS-PL Configuration”, open the “HP Slave AXI Interface” branch and tick the “S AXI HP0 interface” to enable it. Then click OK.



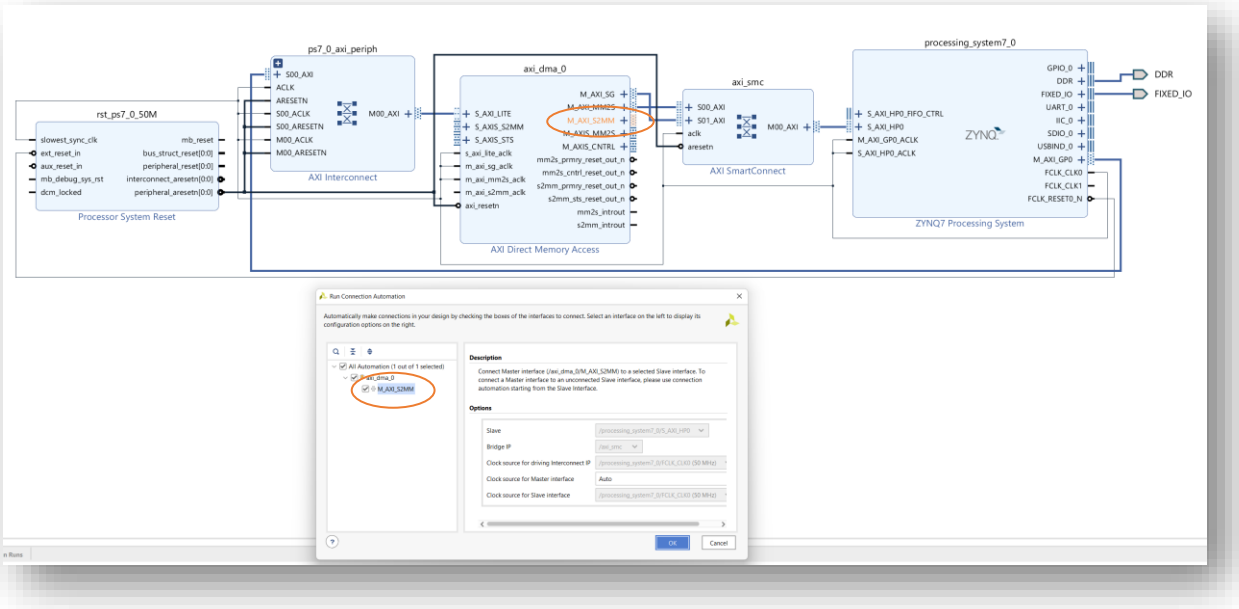
3. The high-performance AXI slave ports should now be visible in the block diagram, and designer assistance should be available. Click the “Run Connection Automation” link and select **/processing_system7_0/S_AXI_HP0** from the drop-down menu. Click “OK”.



4. Designer assistance should again be available, click the “Run Connection Automation” link and select **/axi_dma_0/M_AXI_SG** from the drop-down menu. Click “OK”.



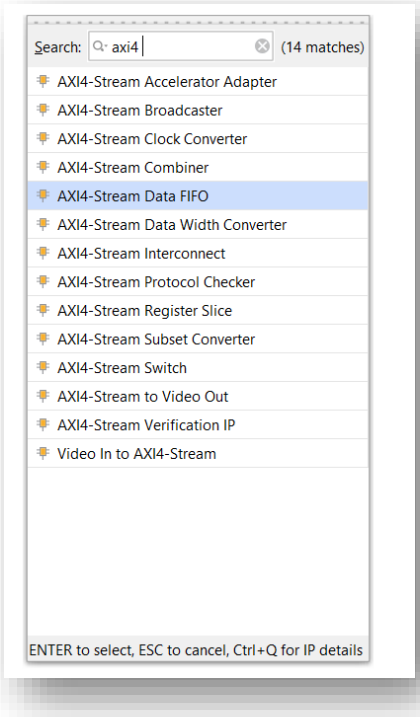
5. Designer assistance should still be available, click the “Run Connection Automation” link and select **/axi_dma_0/M_AXI_S2MM** from the drop-down menu. Click “OK”.



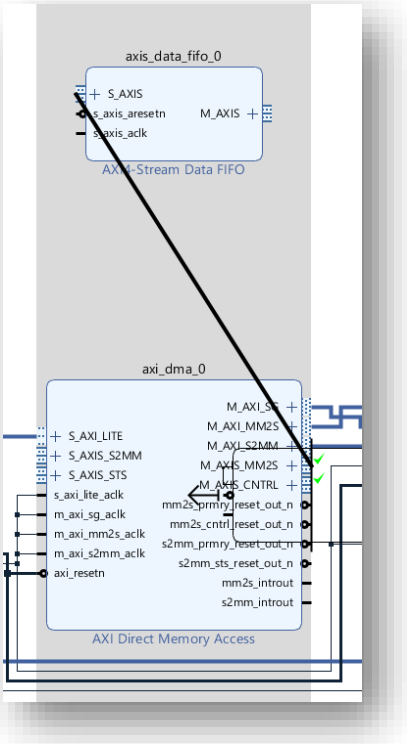
6. Now all the memory mapped AXI buses are connected to the DMA. Now we only must connect the AXI streaming buses to our loopback FIFO and connect the DMA interrupts.

3, Add the FIFO

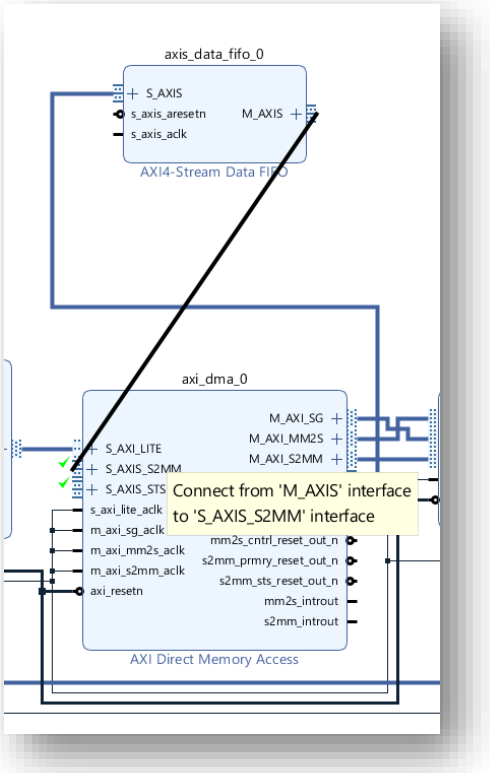
1. Click the “Add IP” icon and double click “AXI4-Stream Data FIFO” from the catalog.



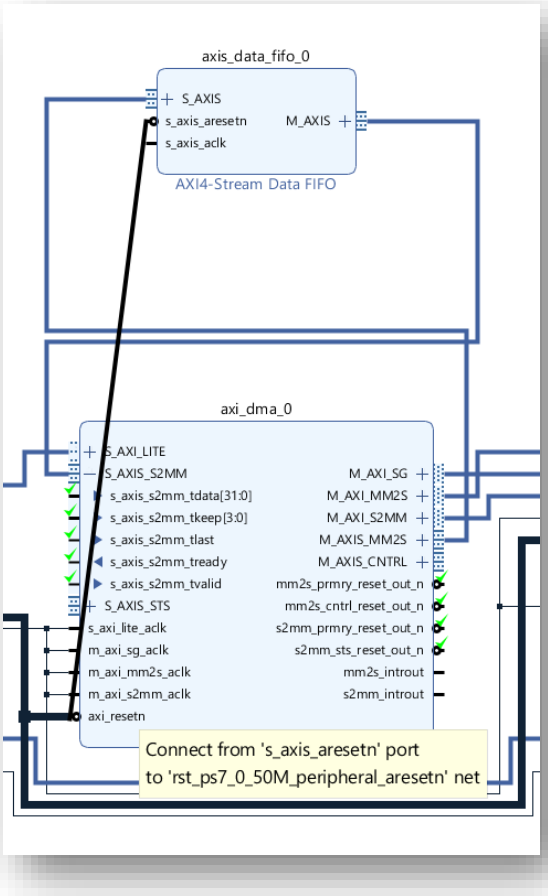
2. The FIFO should be visible in the block diagram. Now we must connect the AXI-streaming buses to those of the DMA. Click the **S_AXIS** port on the FIFO and connect it to the **M_AXIS_MM2S** port of the DMA.



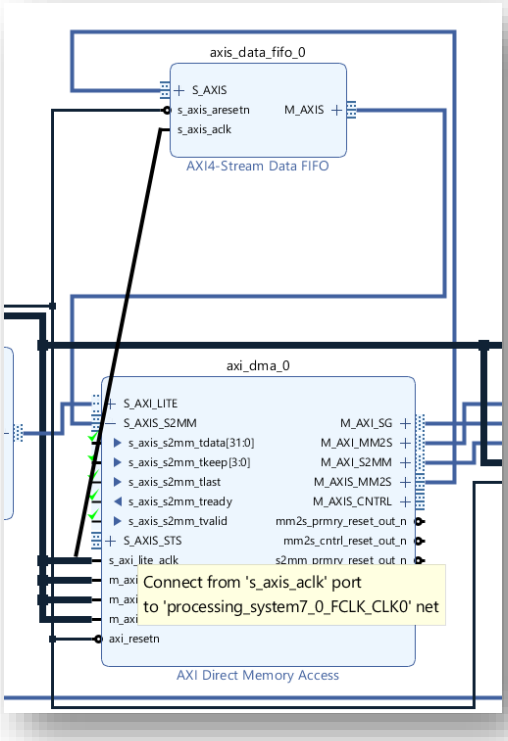
3. Then connect the **M_AXIS** port on the FIFO and connect it to the **S_AXIS_S2MM** port of the DMA.



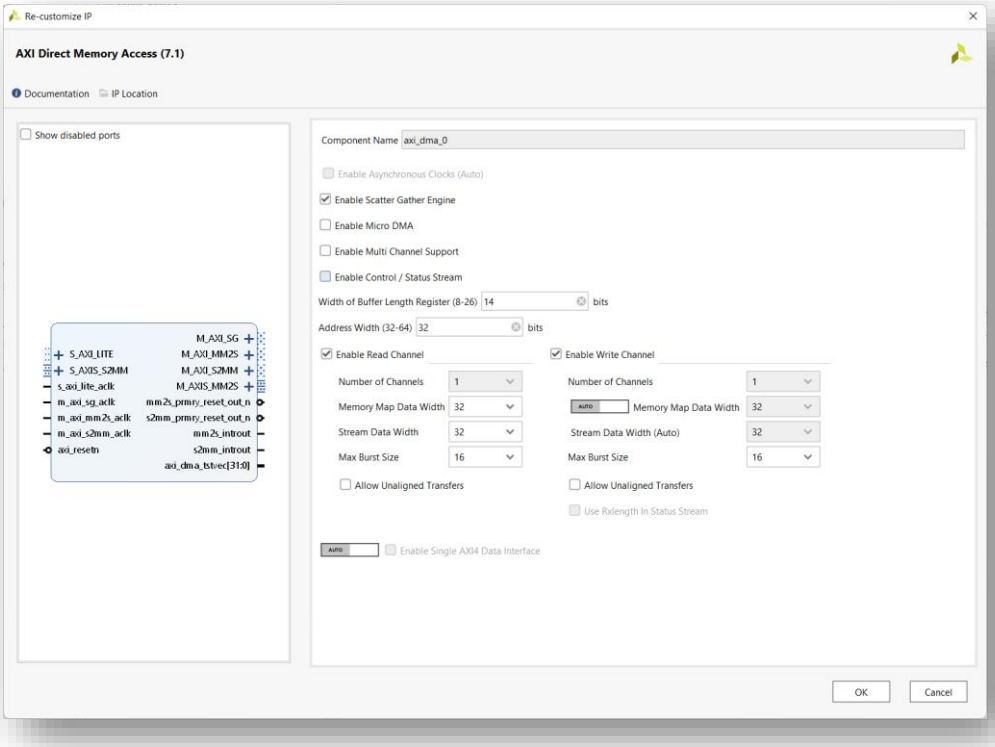
4. Now we must connect the FIFO clock and reset. Click the **s_axis_aresetn** port of the FIFO and connect it to the **axi_resetn** port of the DMA.



5. Click the **s_axis_aclk** port of the FIFO and connect it to the **s_axi_lite_aclk** port of the DMA.



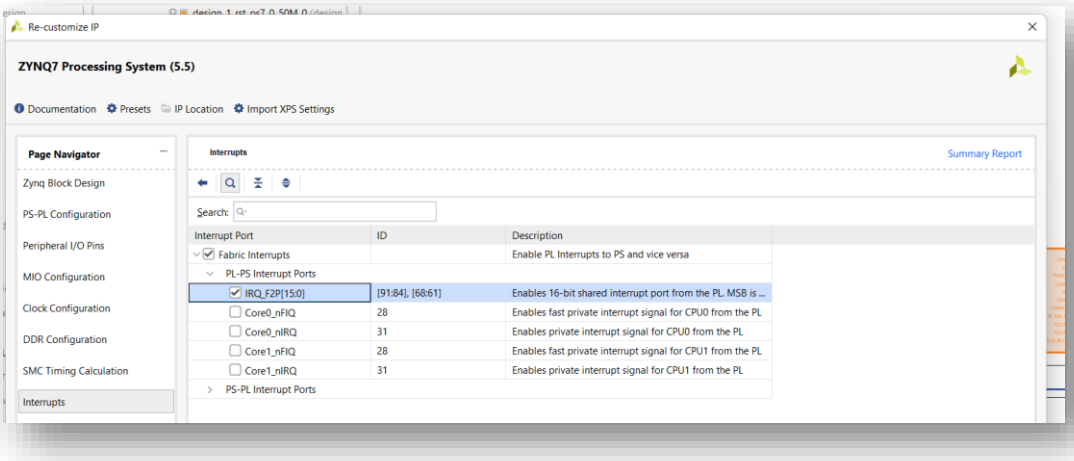
6. In our design, we won't need the AXI-Streaming status and control ports which are used to transmit extra information alongside the data stream. You might use them if you were connecting to the AXI Ethernet core or a custom IP that made use of them. In the block diagram, double click the AXI DMA block. Un-tick the "Enable Control / Status Stream" option and click OK.



4, Enable interrupt from the DMA

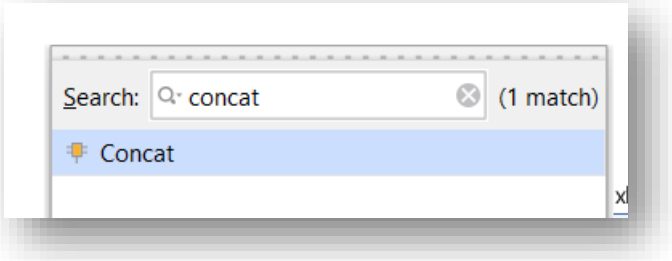
Our software application will test the DMA in polling mode, but to be able to use it in interrupt mode, we need to connect the interrupts **mm2s_introut** and **s2mm_introut** to the Zynq PS.

1. First, we must enable interrupts from the PL. Double click the Zynq block and select the Interrupts tab.

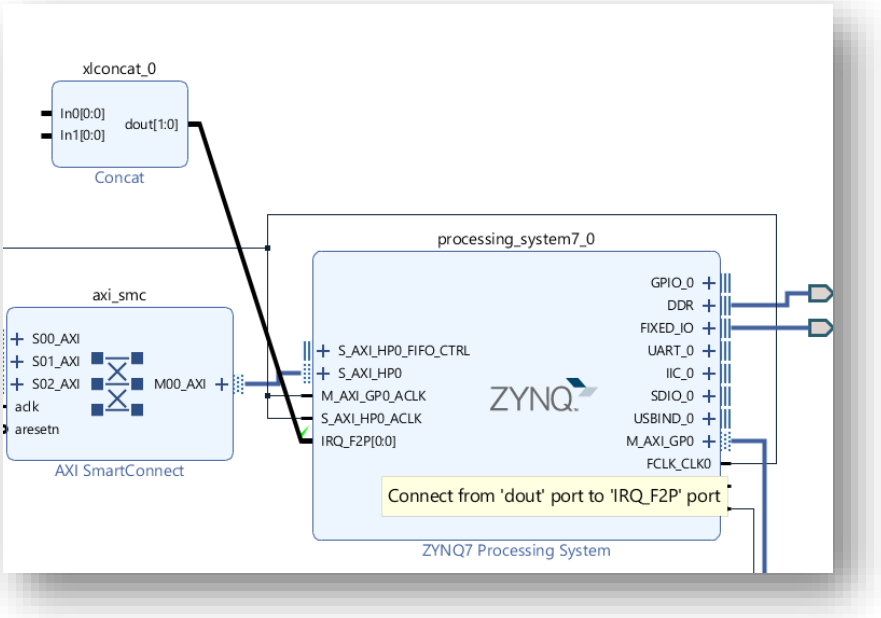


2. Tick "Fabric Interrupts" and IRQ_F2P[15:0] to enable them, and click OK.

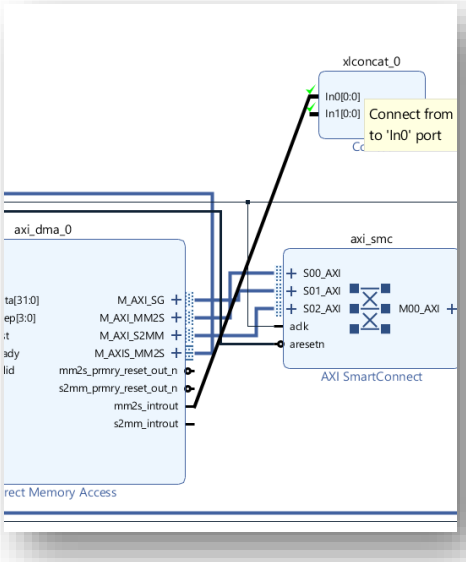
3. Click the “Add IP” icon and double-click “Concat” from the catalog.



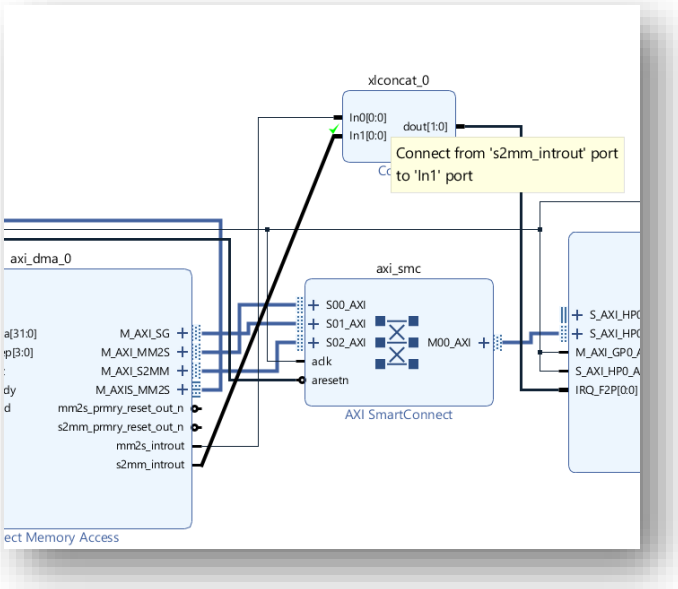
4. Connect the **dout** port of the Concat to the **IRQ_F2P** port of the Zynq PS.



5. Connect the **mm2s_introut** port of the DMA to the **In0** port of the Concat.

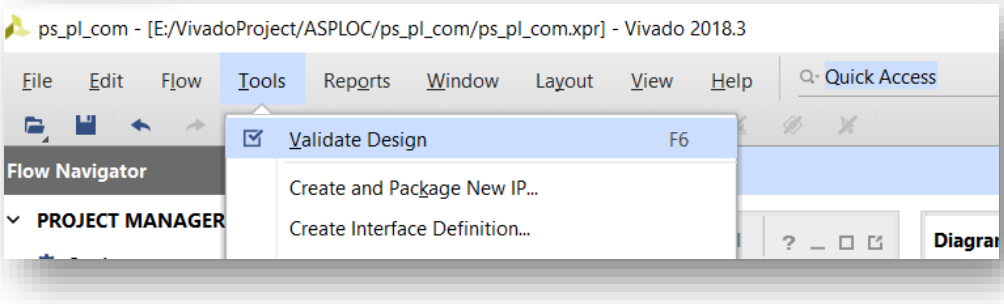


6. Connect the **s2mm_introut** port of the DMA to the **In1** port of the Concat.

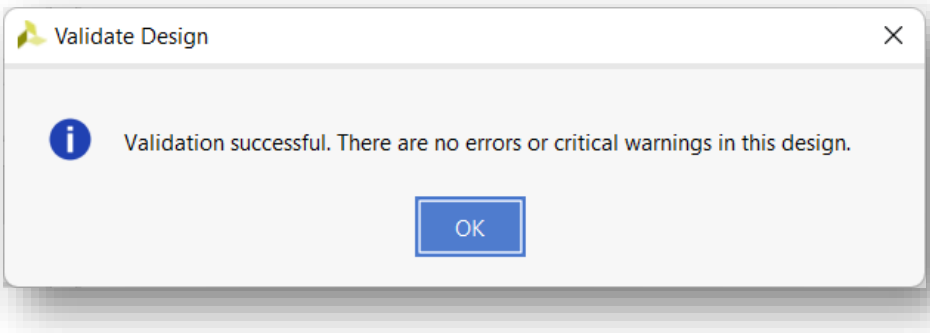


6, Validate and build the design

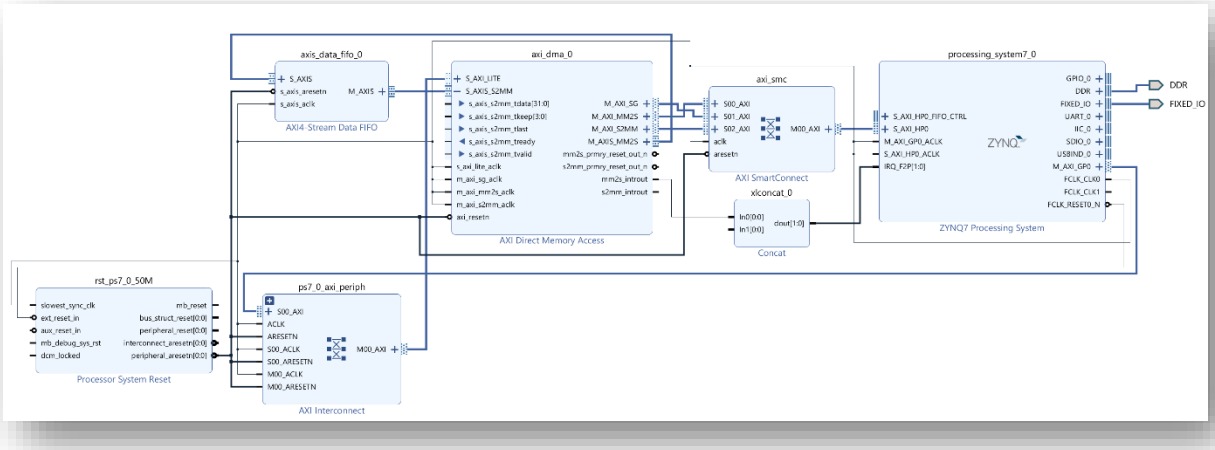
1. From the menu select Tools->Validate Design.



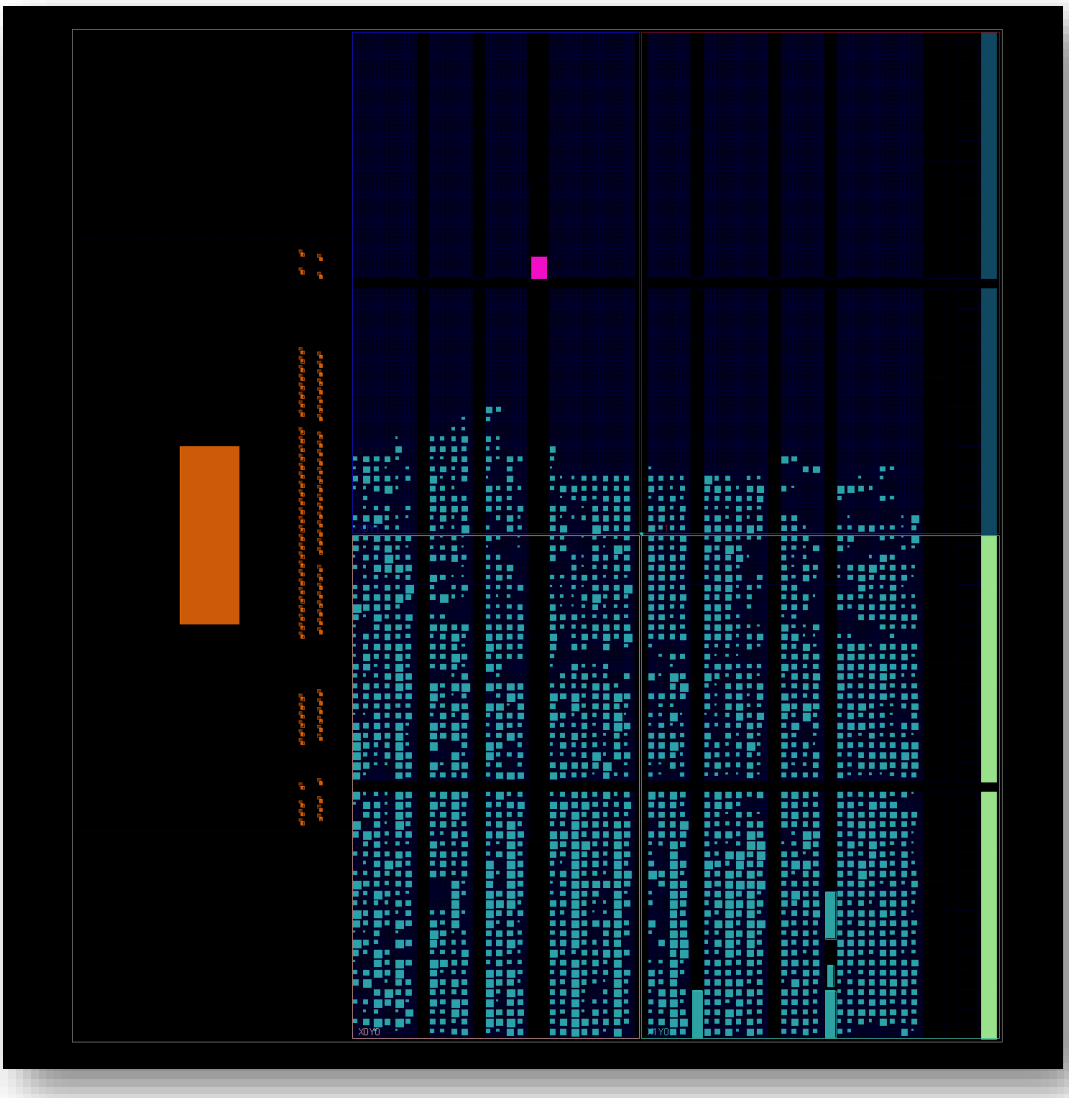
2. You should get this message saying that validation was successful.



3. Our block diagram now looks like this.



4. In the Flow Navigator, click “Generate Bitstream”. It may take a few minutes to generate the bit stream. Below is the LUT utilization overview for the above design. Lightblue boxes indicate busy LUTs while dark ones the free LUTs, which you can use to implement your own design.

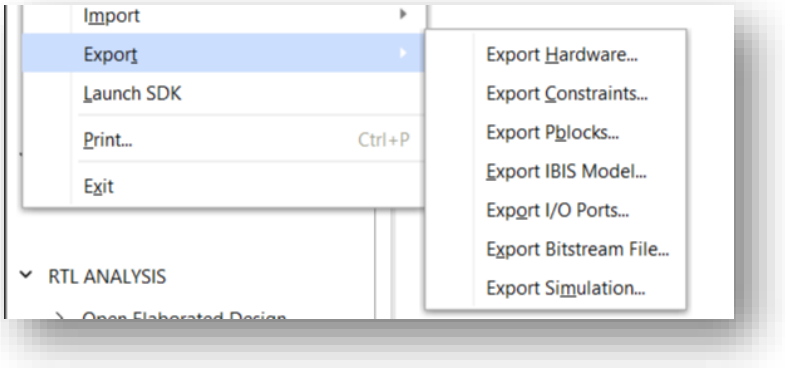


PART 2 – Programming the DMA system using C

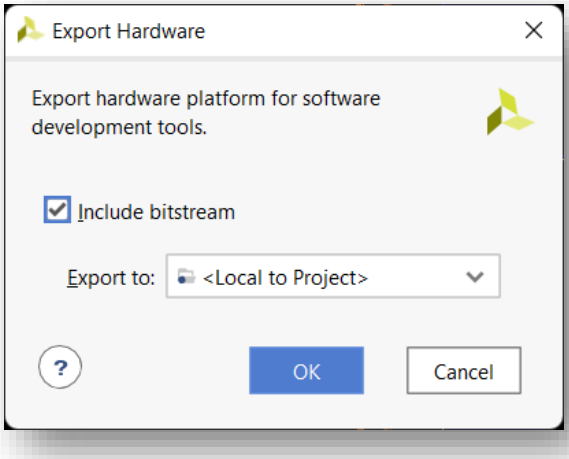
1, Export the hardware design to SDK

Once the bitstream has been generated, we can export our design to SDK where we can develop the software application that will setup a DMA transfer, wait for completion and then verify the loopback.

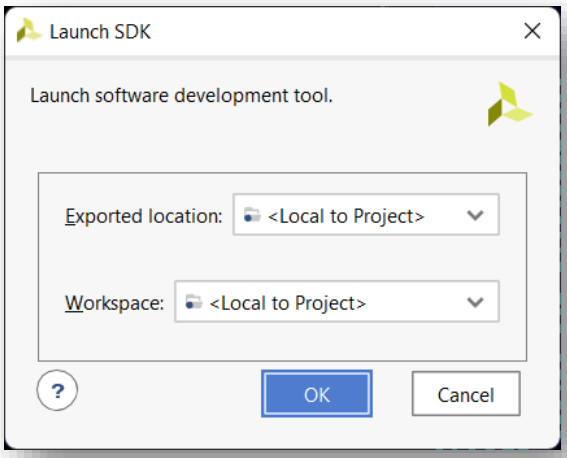
1. In Vivado, from the File menu, select “Export->Export hardware”.

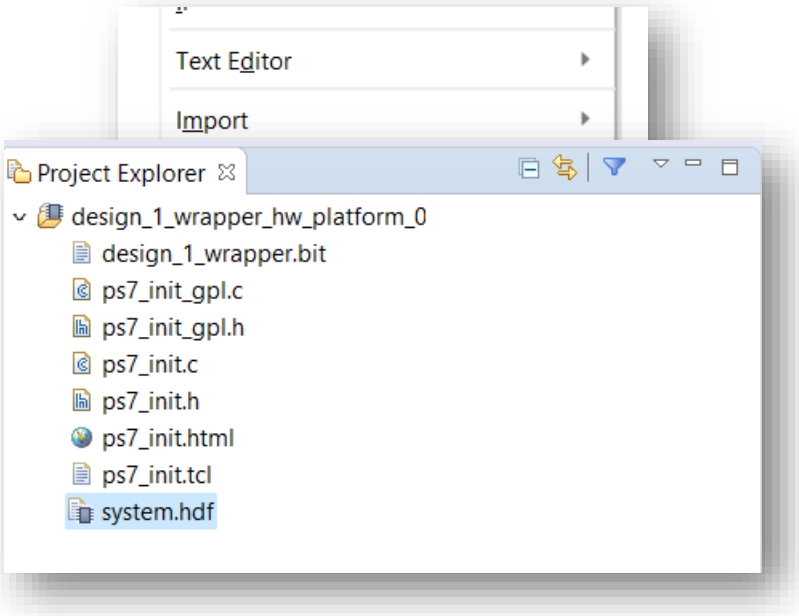


2. In the window that appears, tick “Include bitstream” and click “OK”.



3. Again, from the File menu, select “Launch SDK”.
4. In the window that appears, use the following settings, and click “OK”.

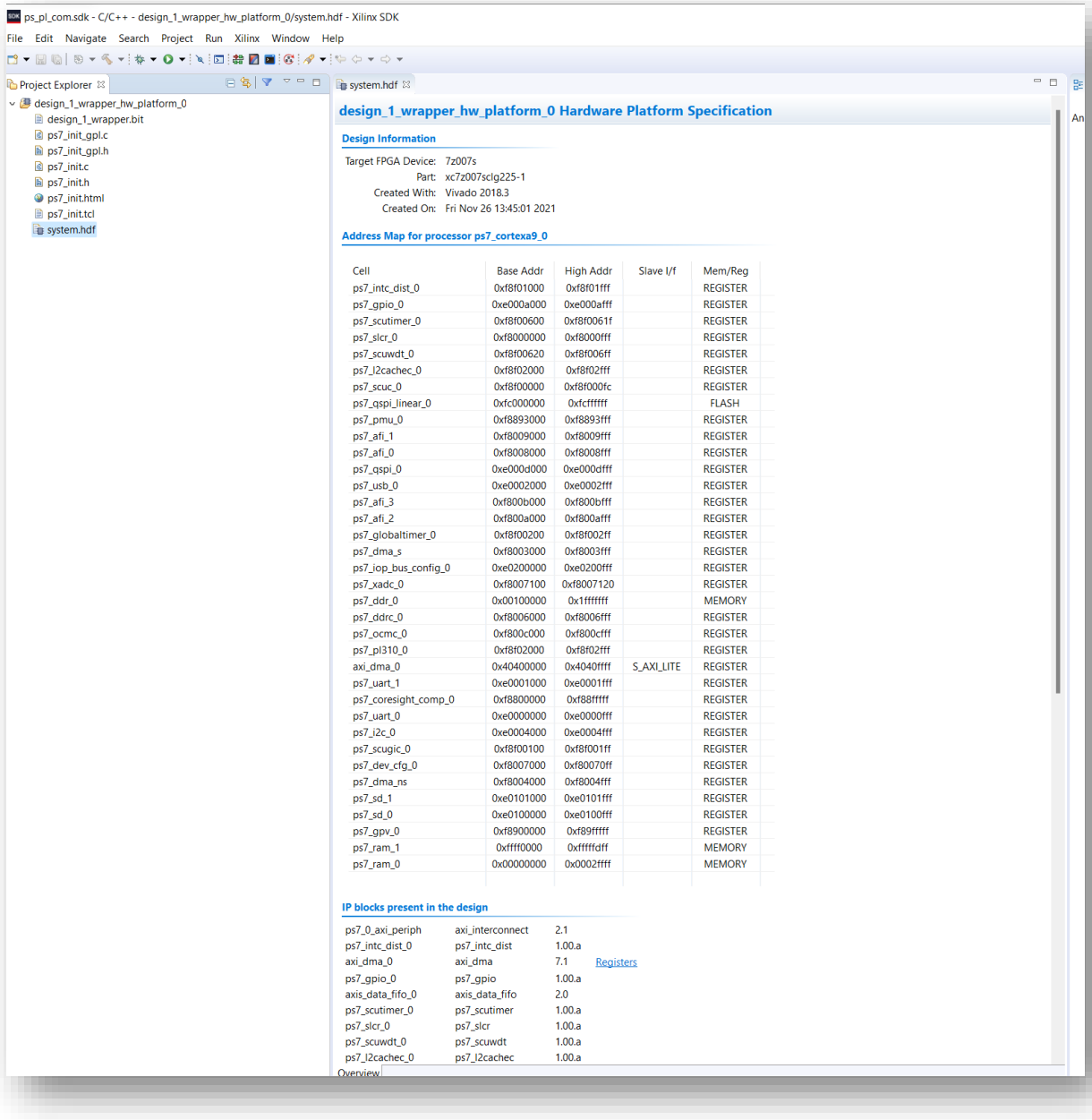




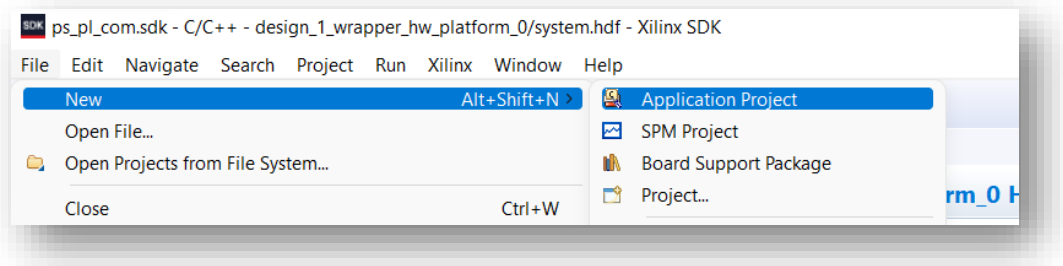
5. At this point, the SDK loads and a hardware platform specification will be created for your design. You should be able to see the hardware specification in the Project Explorer of SDK as shown in the image below.
6. We are now ready to create the software application.

2, Create a software application

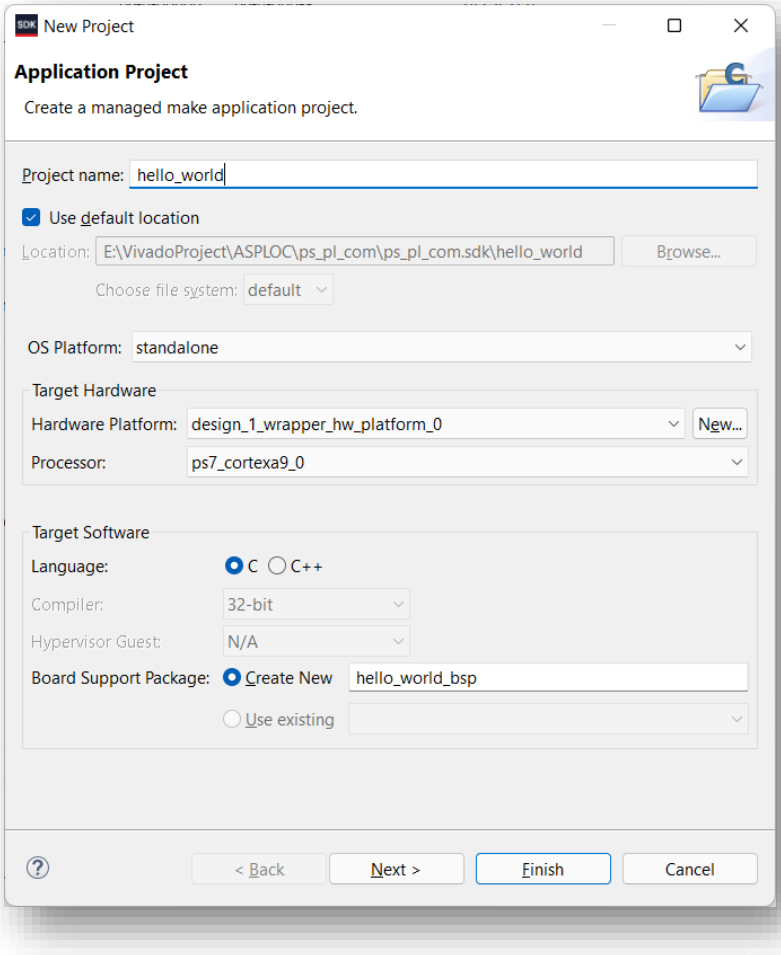
1. At this point, your SDK window should look somewhat like this:



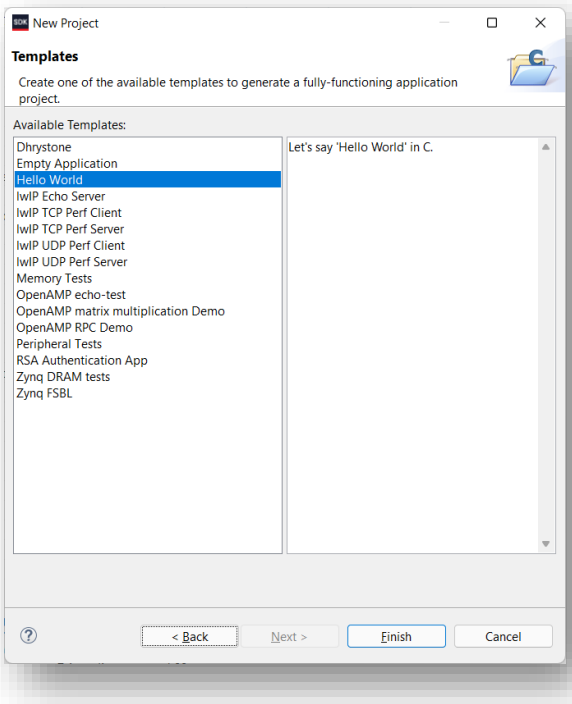
2. To make things easy for us, we'll use the template for the hello world application and then modify it to test the AXI DMA. From the File menu, select New->Application Project.



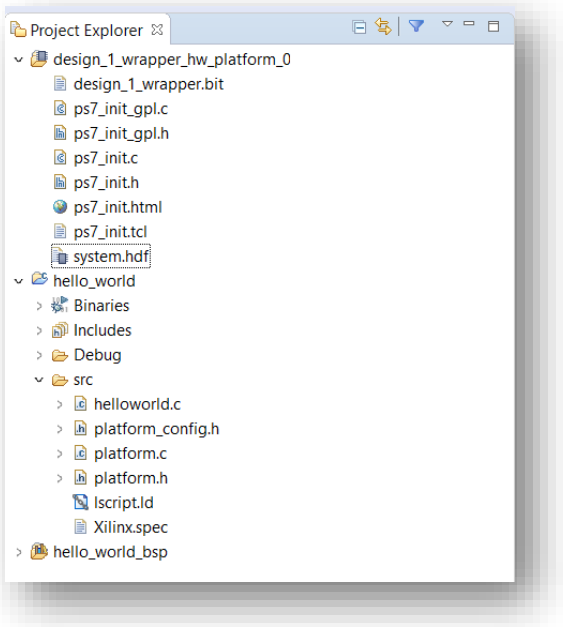
3. In the first page of the New Project wizard, choose a name for the application. I've chosen hello_world. Click "Next".



4. On the templates page, select “Hello World” template and click “Finish”.



5. The SDK will generate a new application which you should find in the Project Explorer as in the image below.

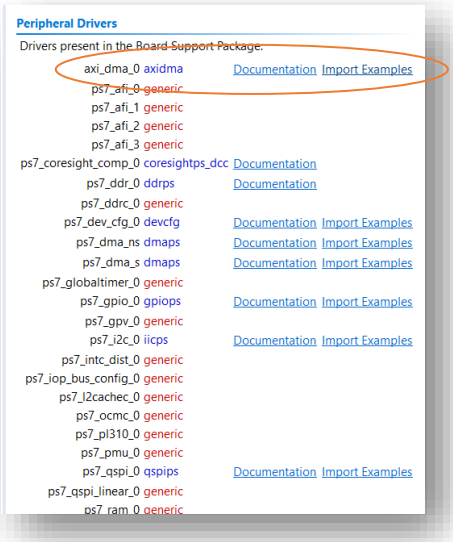


6. The hello_world folder contains the Hello World software application, which we will modify to test our AXI DMA.

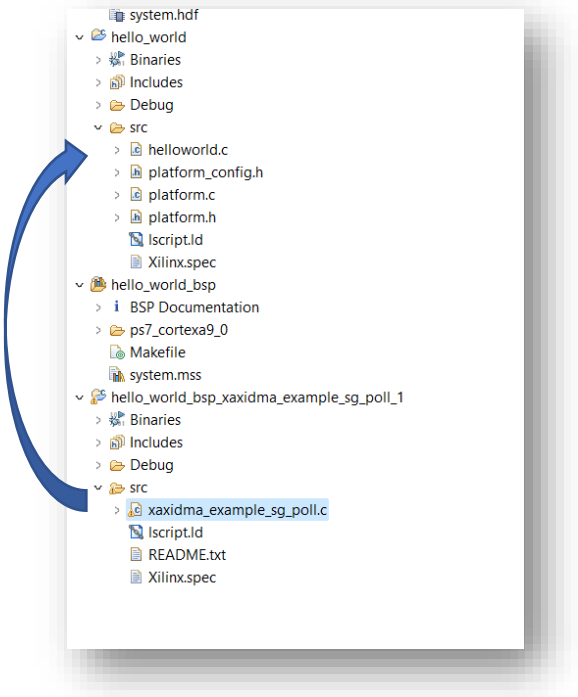
3, Modify the software application

We need to modify the hello world software application to test our DMA. The application source code we are using in this tutorial is one of the many valuable examples provided by Xilinx in the installation files. If you didn't know about those examples, I suggest you check it out every time you start playing with a new IP core.

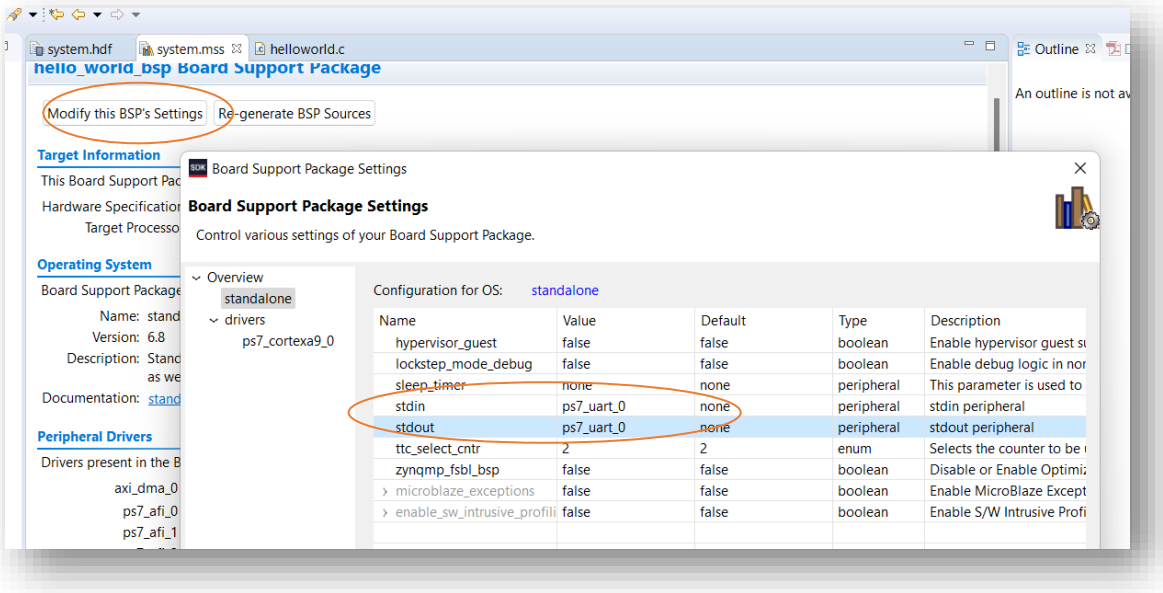
1. From the Project Explorer, open the hello_world/src folder. Open the "helloworld.c" source file.
2. Replace all the code in this file with the code that you will find in the Vivado example project for AXI DMA.



3. Copy the contents of “xaxidma_example_sg_polll.c” into “helloworld.c”.

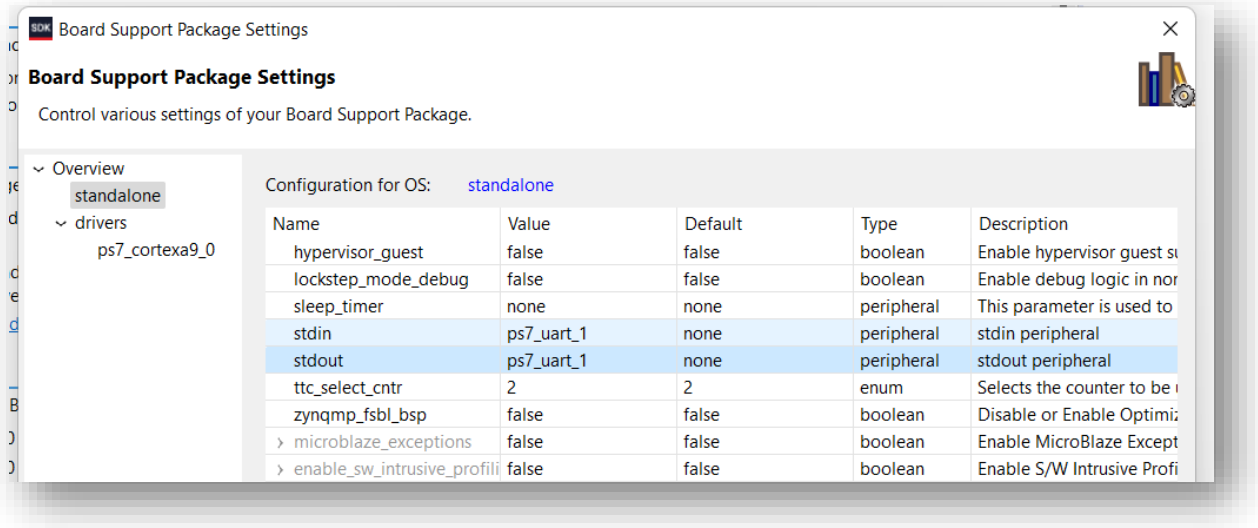


4. Compile the project and launch it on the board. If everything is successful, the output message “Successfully ran AXI DMA SG Polling Example” will appear in the com port.
5. Note that by default the “hello world” project template initiate stdin/stdout to ps7_uart_0.

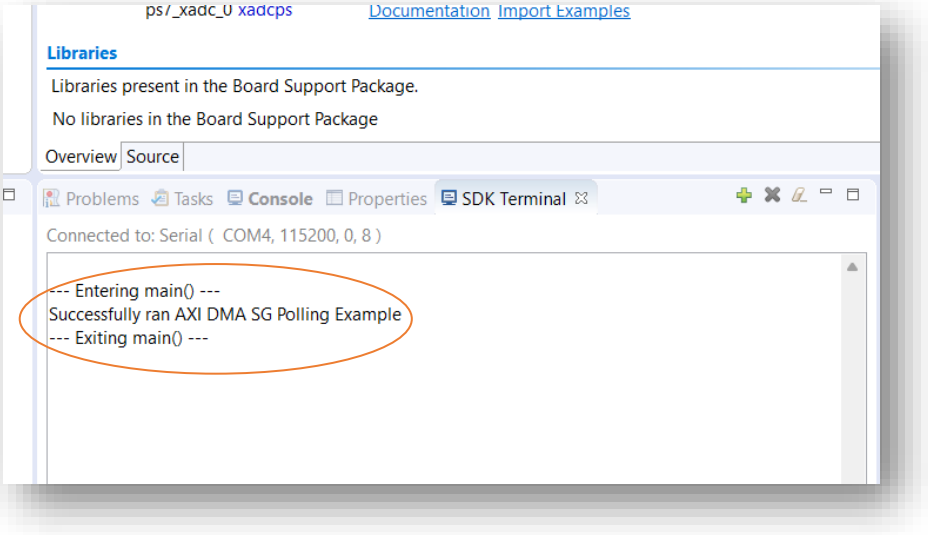


6. However, if we go back to Vivado and open up the Zynq7000 configuration view for peripheral I/O, we can see that in our system, by default, UART 0 is used for EMIO but UART 1 is used for communication port UART 1.

7. Modify the bsp file so that stdin/stdout are pointing to UART 1 instead of UART 0.



8. Ater the modification you should see the output correctly in the COM port.



9. If you see this message, then everything should be set correctly in the whole HW/SW platform.