FPGAdeveloper

Using the AXI DMA in Vivado

by Jeff Johnson | Aug 6, 2014 | Vivado | 19 comments

14 Votes

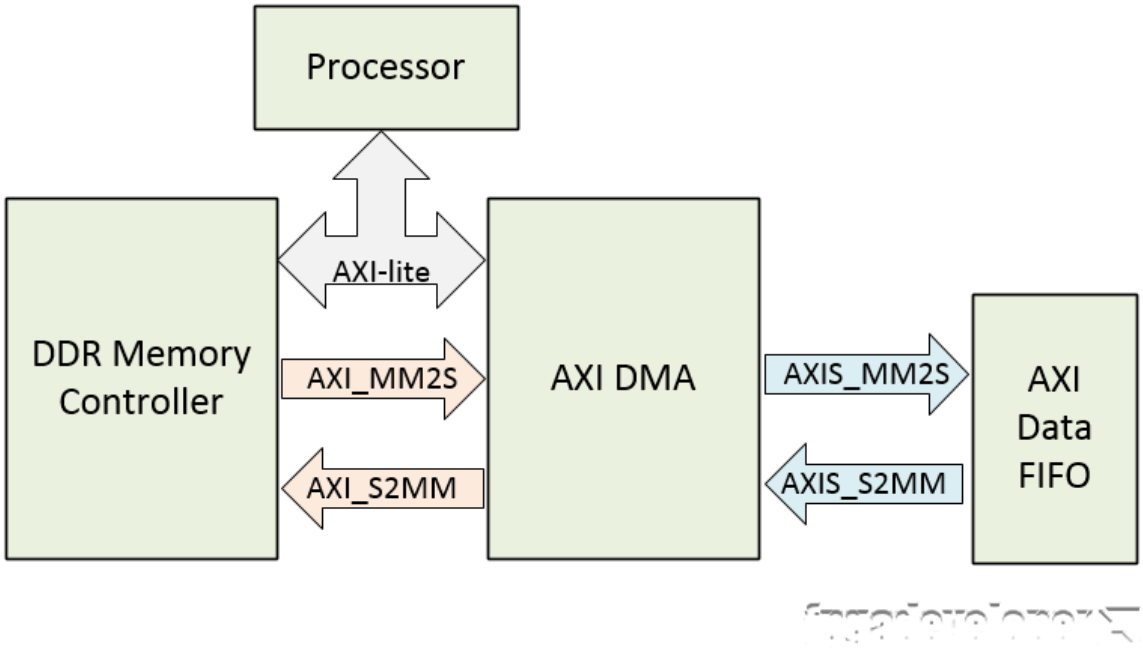
In a previous tutorial I went through how to use the AXI DMA Engine in EDK, now I'll show you how to use the AXI DMA in Vivado. We'll create the hardware design in Vivado, then write a software application in the Xilinx SDK and test it on the MicroZed board (source code is shared on Github for the MicroZed and the ZedBoard, see links at the bottom).

What is DMA?

DMA stands for Direct Memory Access and a DMA engine allows you to transfer data from one part of your system to another. The simplest usage of a DMA would be to transfer data from one part of the memory to another, however a DMA engine can be used to transfer data from any data producer (eg. an ADC) to a memory, or from a memory to any data consumer (eg. a DAC).

Tutorial overview

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 an ADC/DAC FMC, 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.



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.

Requirements

Before following this tutorial, you will need to do the following:

- Vivado 2014.2
- MicroZed
- Platform Cable USB II (or equivalent JTAG programmer)

Start from the base project

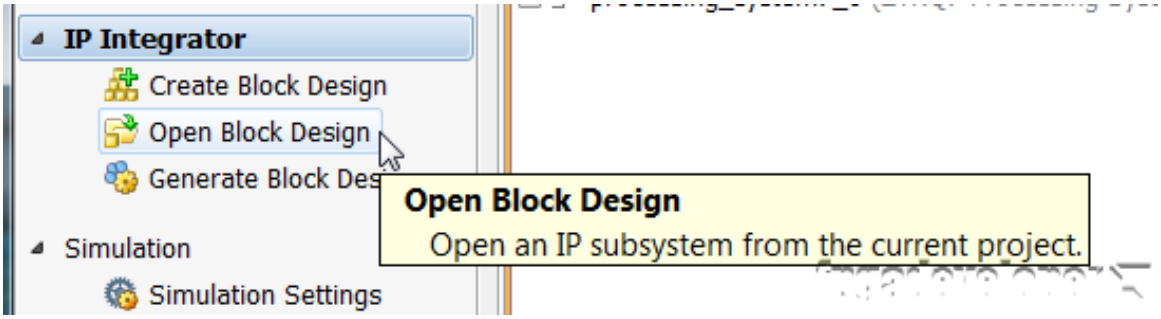
We'll start this tutorial with the base system project for the MicroZed that you can access here:

Base system project for the MicroZed

Add the AXI DMA

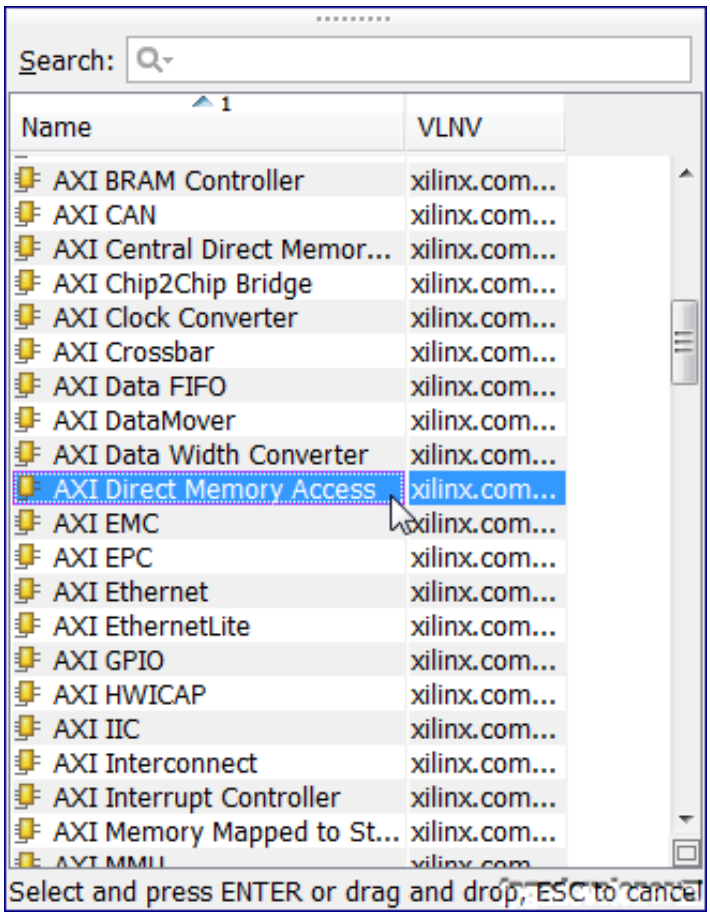
1. Open the base project in Vivado.

2. In the Flow Navigator, click 'Open Block Design'.



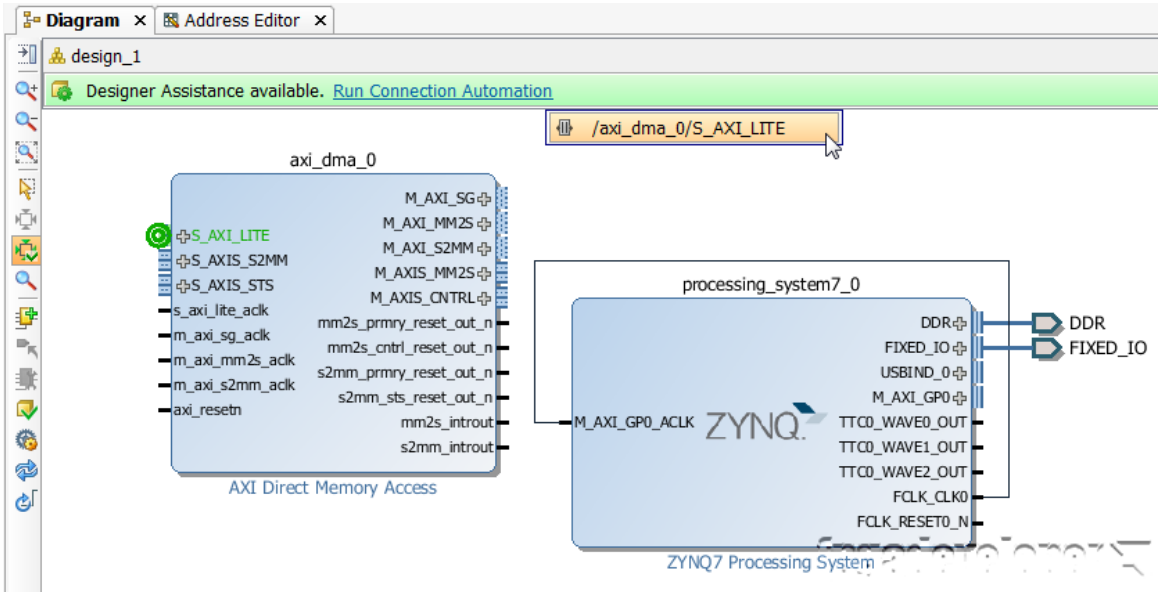
3. The block diagram should open and you should only have the Zynq PS in the design.

4. Click the 'Add IP' icon and double click 'AXI Direct Memory Access' from the catalog.

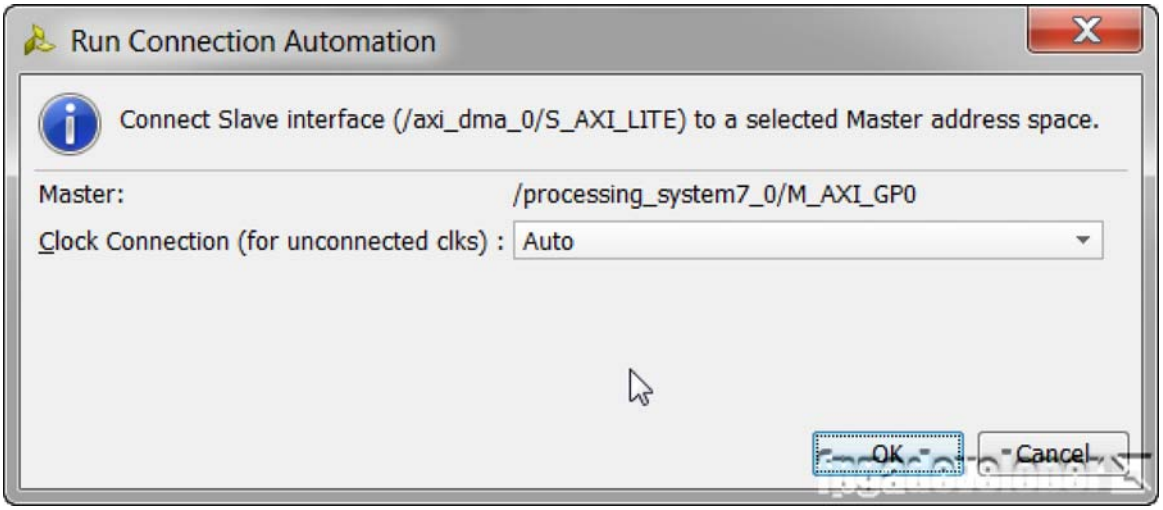


Connect the Memory-mapped AXI buses

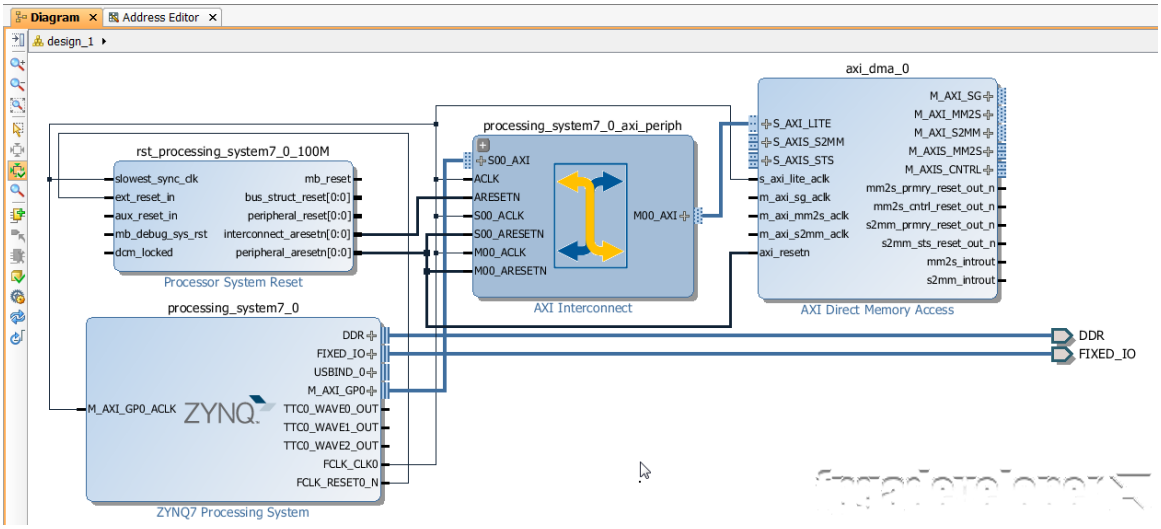
1. The DMA block should appear and designer assistance should be available. Click the 'Run Connection Automation' link and select '/axi_dma_0/S_AXI_LITE' from the drop-down menu.



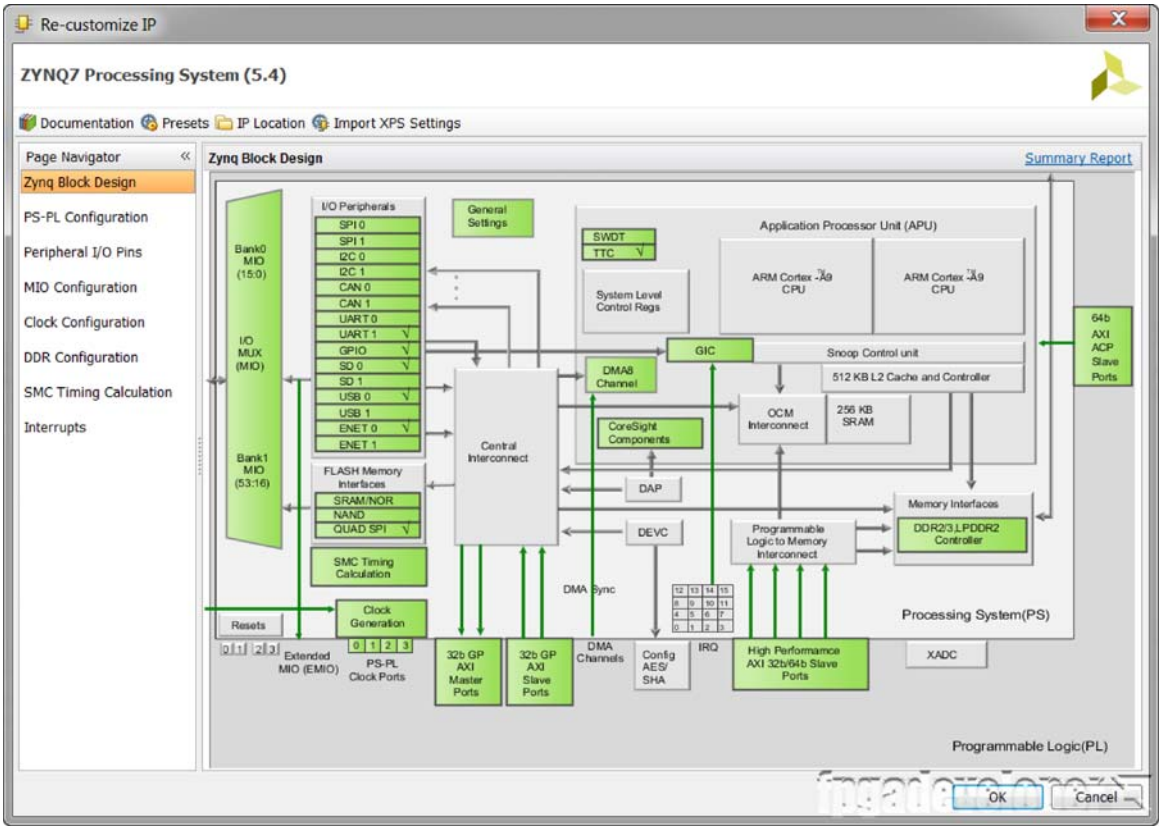
2. Click 'OK' in the window that appears. Vivado will connect the AXI-lite bus of the DMA to the General Purpose AXI Interconnect of the PS.



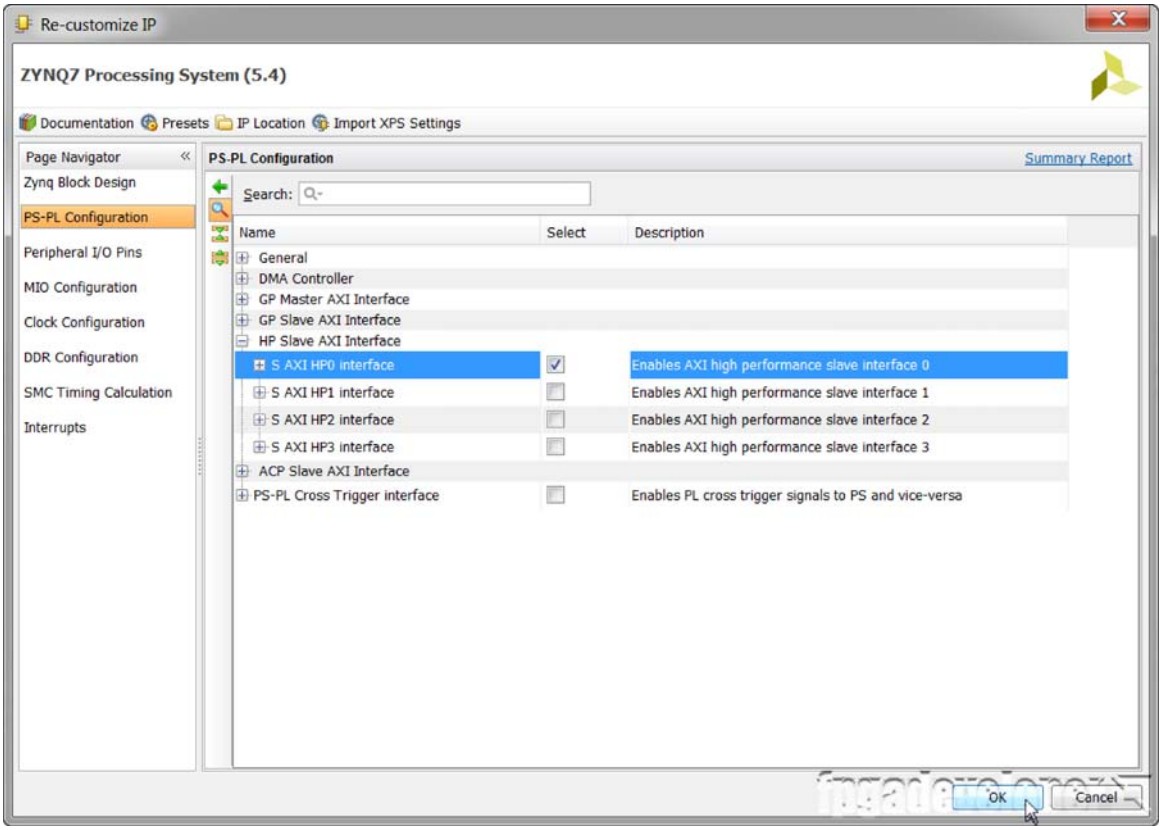
3. Your block diagram should now look like this :



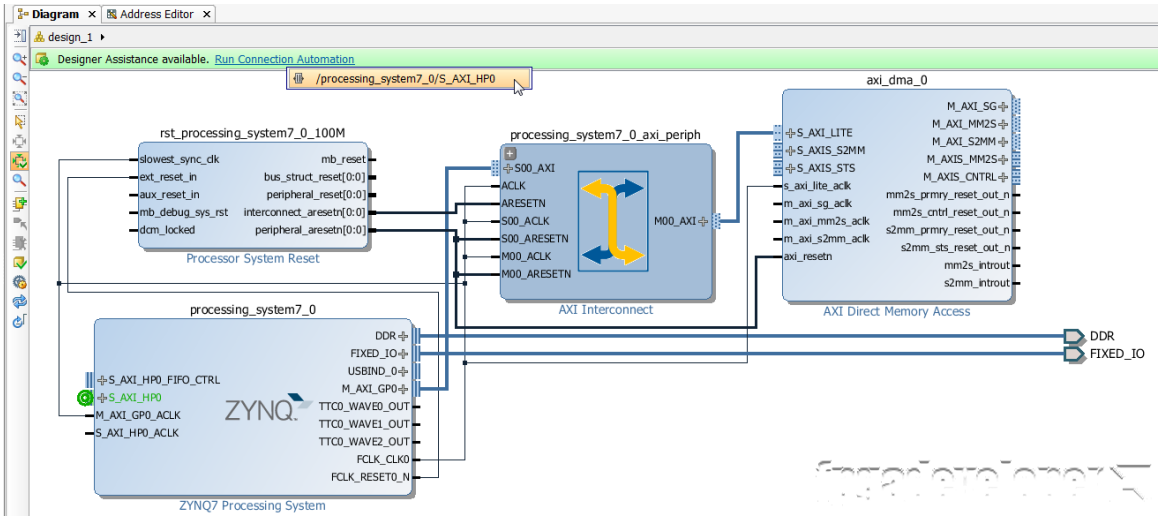
4. Now we need to connect AXI buses M_AXI_SG, M_AXI_MM2S and M_AXI_S2MM of the DMA to a high performance AXI slave interface on the PS. Our PS doesn't seem to have a high-performance AXI slave interface, so we need to change the Zynq configuration to enable one. Double click on the Zynq block.



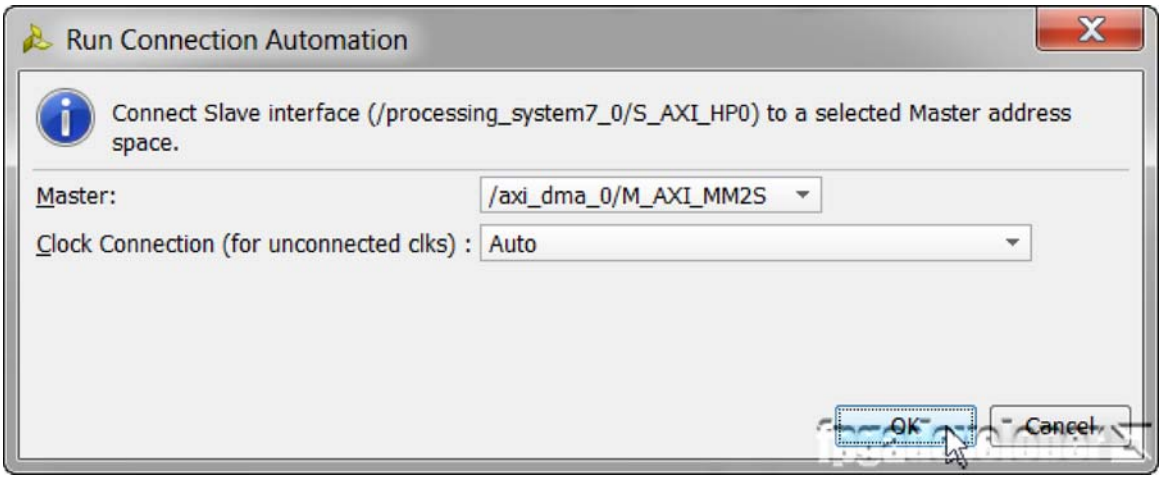
5. Select ‘PS-PL Configuration’, open the ‘HP Slave AXI Interface’ branch and tick the ‘S AXI HP0 interface’ to enable it. Then click OK.



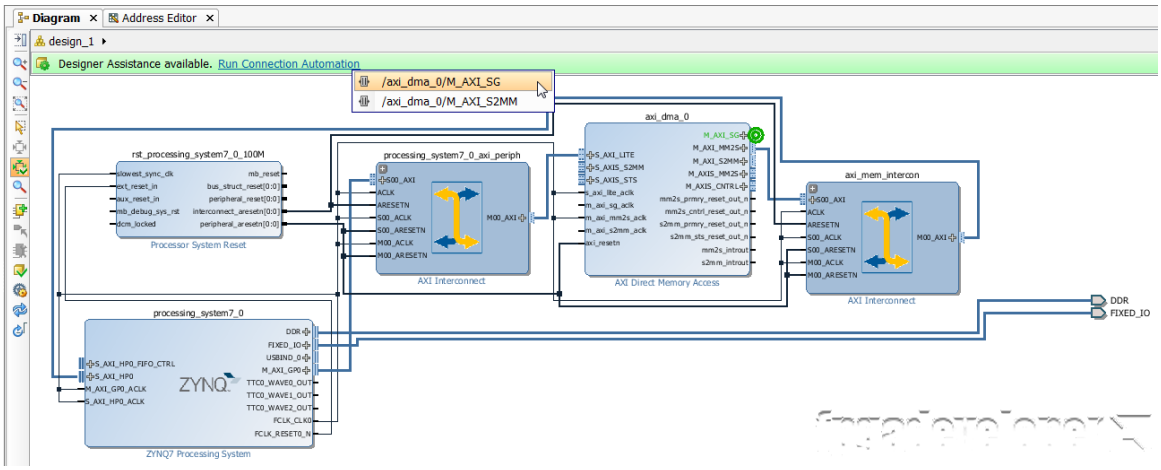
6. The high-performance AXI slave ports should now be visible in the block diagram, and designer assistance should be available. Click the ‘Run Connection Automation’ link and select ‘/processing_system7_0/S_AXI_HP0’ from the drop-down menu.



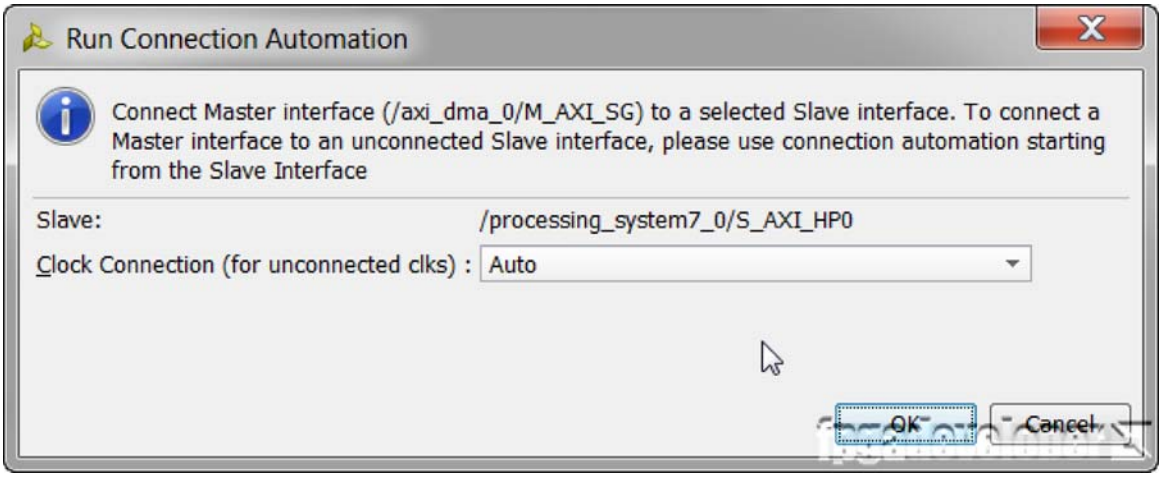
7. In the window that appears, make sure that Vivado intends to connect it to the DMA and click OK.



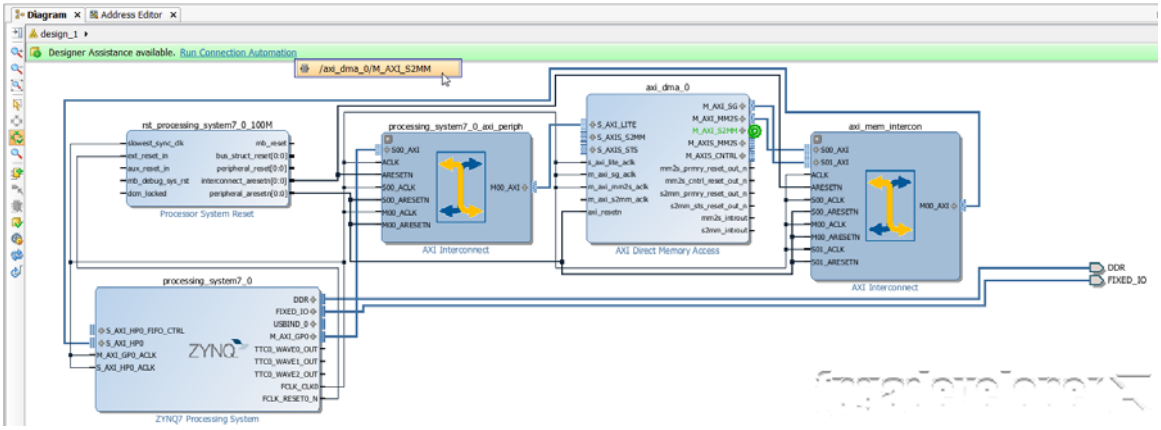
8. Designer assistance should again be available, click the 'Run Connection Automation' link and select '/axi_dma_0/M_AXI_SG' from the drop-down menu.



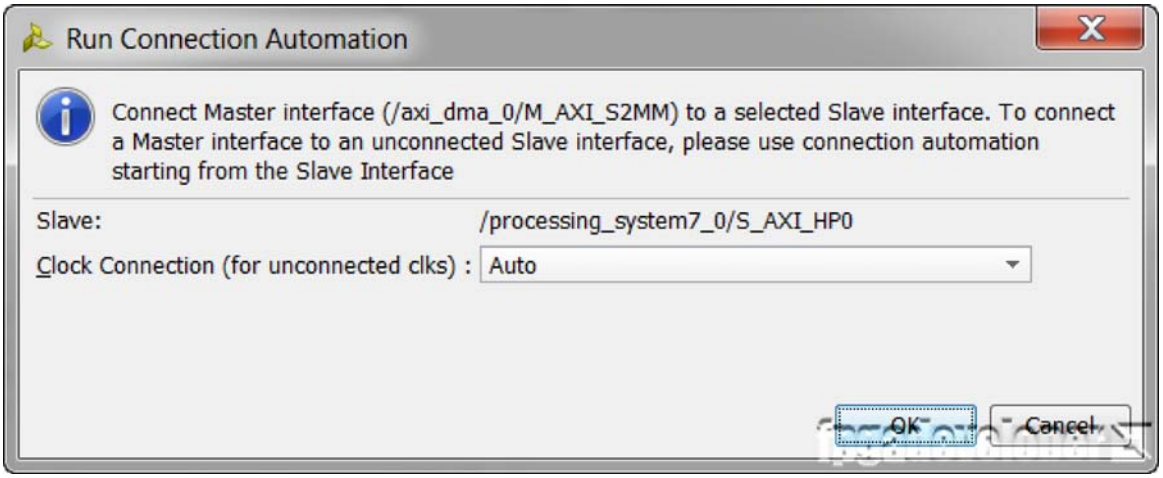
9. In the window that appears, click OK.



10. 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.



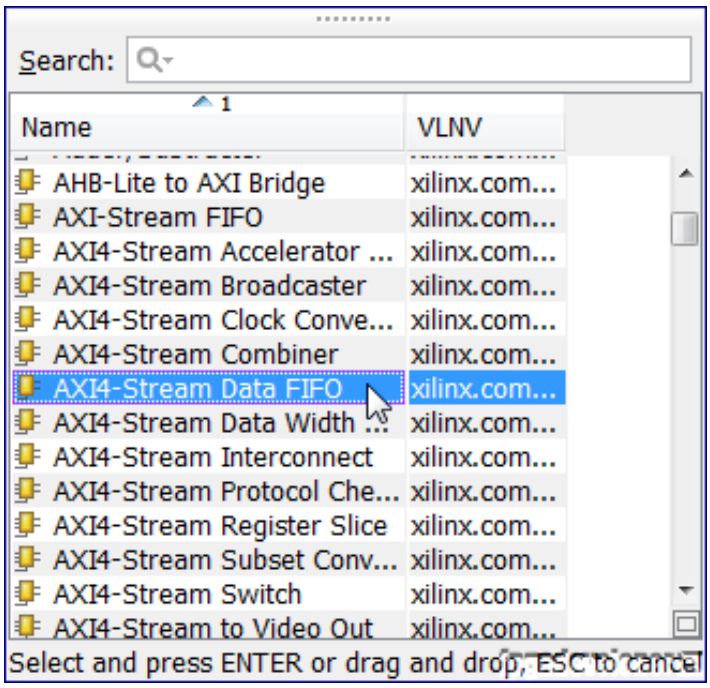
11. In the window that appears, click OK.



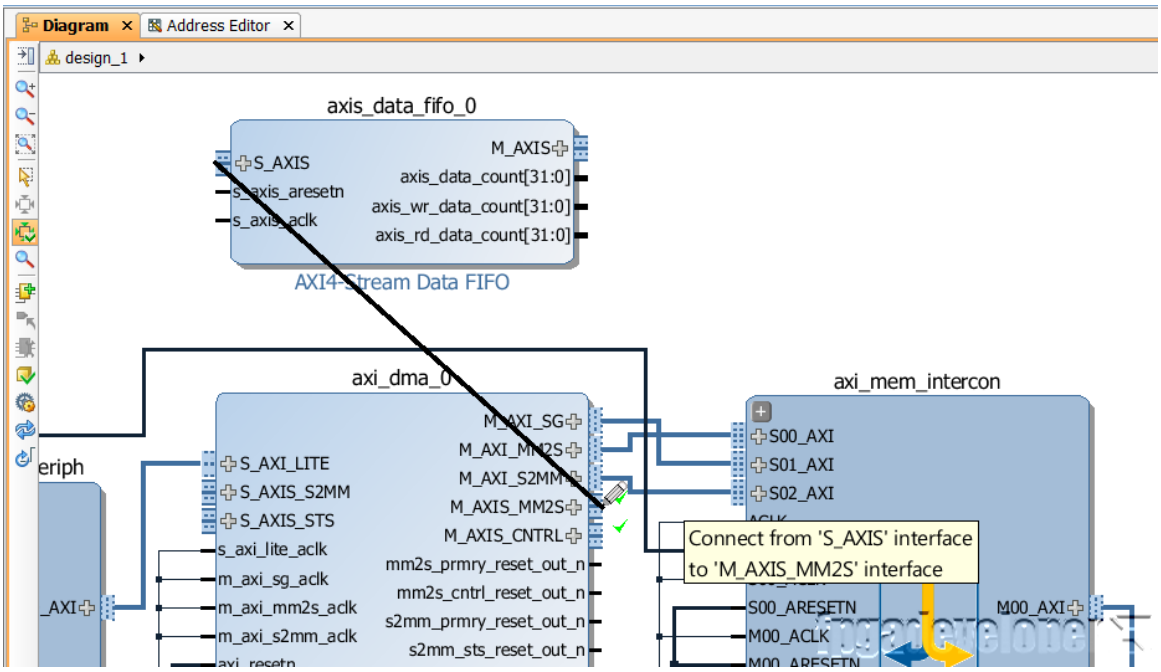
Now all the memory-mapped AXI buses are connected to the DMA. Now we only have to connect the AXI streaming buses to our loopback FIFO and connect the DMA interrupts.

Add the FIFO

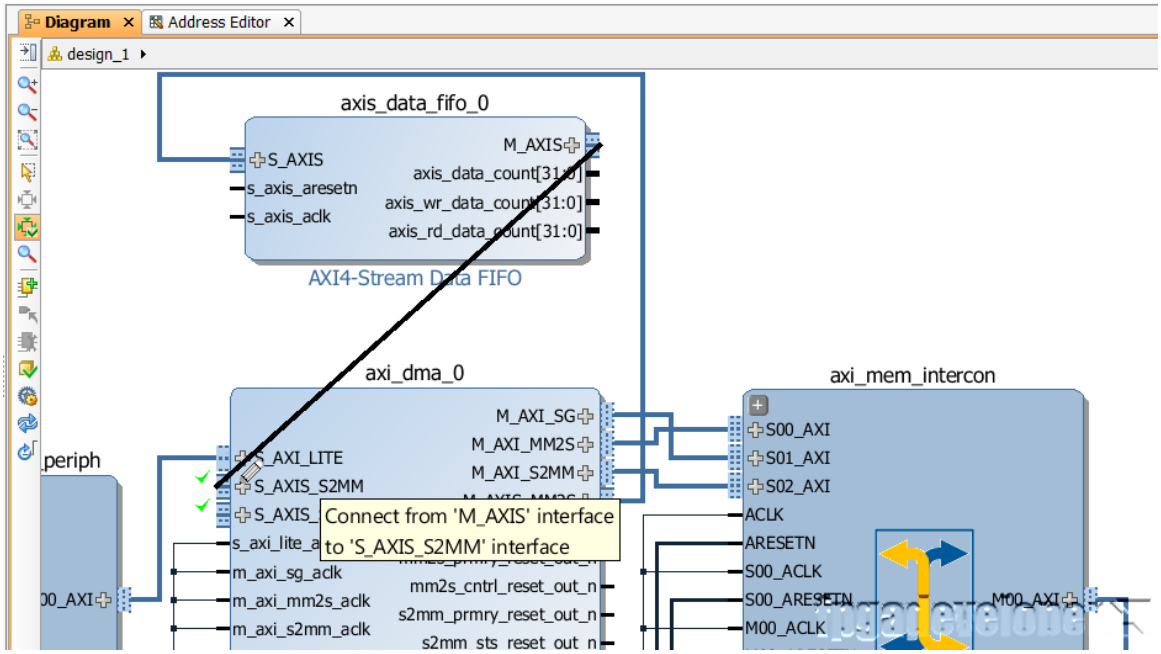
1. Click the 'Add IP' icon and double click 'AXI4-Stream Data FIFO' from the catalog.



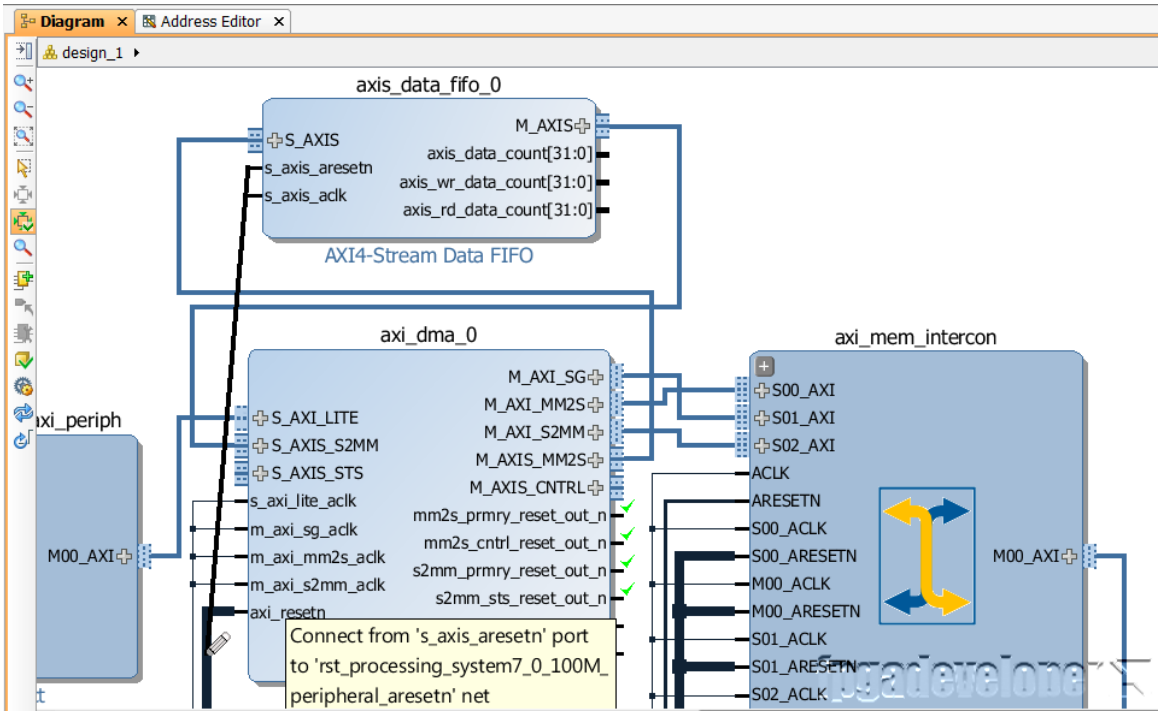
2. The FIFO should be visible in the block diagram. Now we must connect the AXI-streaming buses to those of the DMA. Click the 'S_AXIS' port on the FIFO and connect it to the 'M_AXIS_MM2S' port of the DMA.



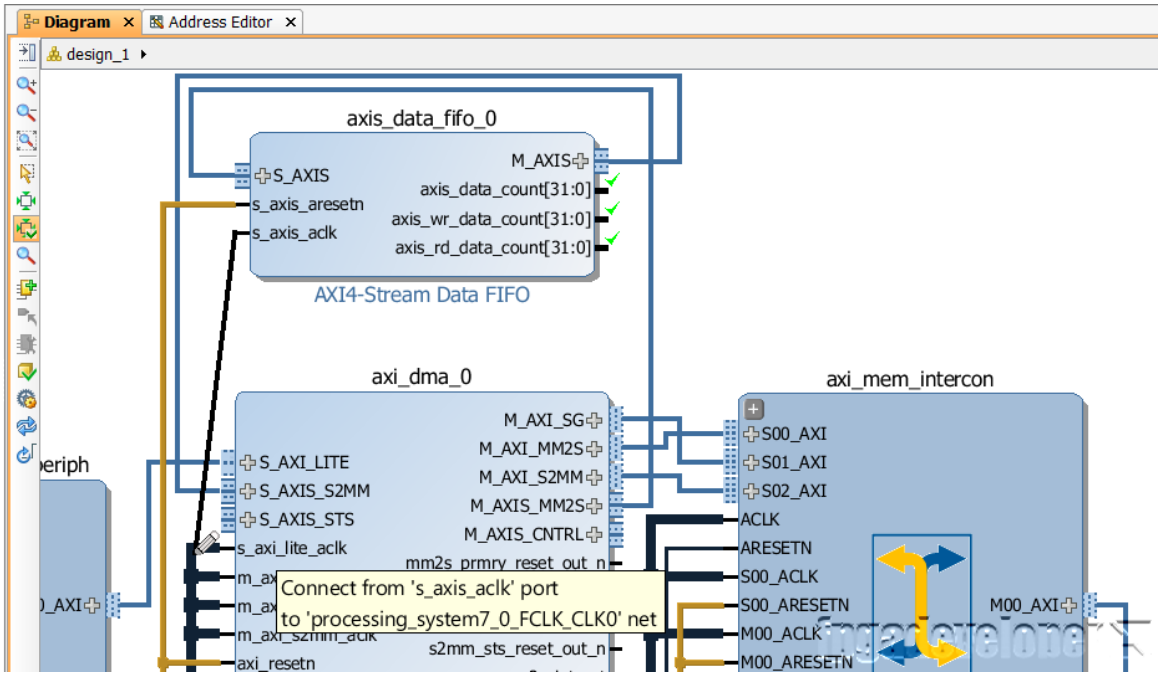
3. Then connect the 'M_AXIS' port on the FIFO and connect it to the 'S_AXIS_S2MM' port of the DMA.



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.



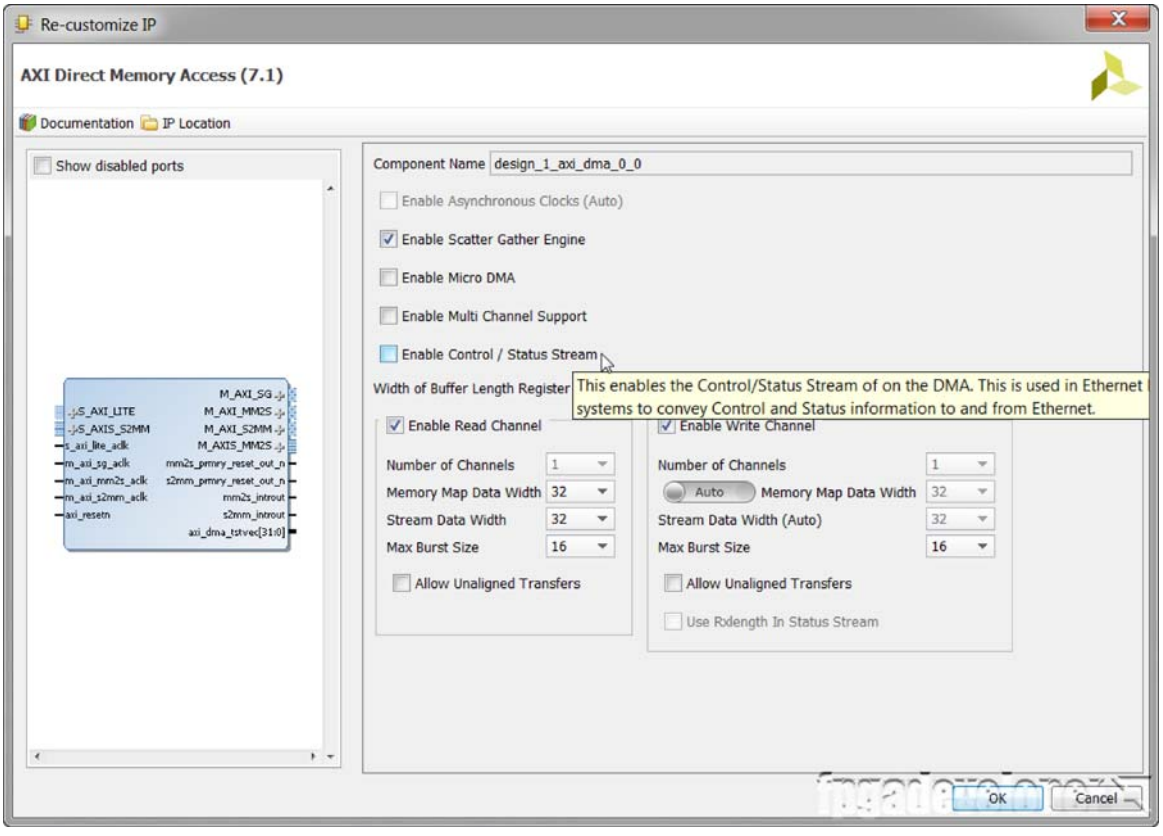
5. Click the 's_axis_aclk' port of the FIFO and connect it to the 's_axi_lite_aclk' port of the DMA.



Remove the AXI-Streaming status and control ports of the DMA

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.

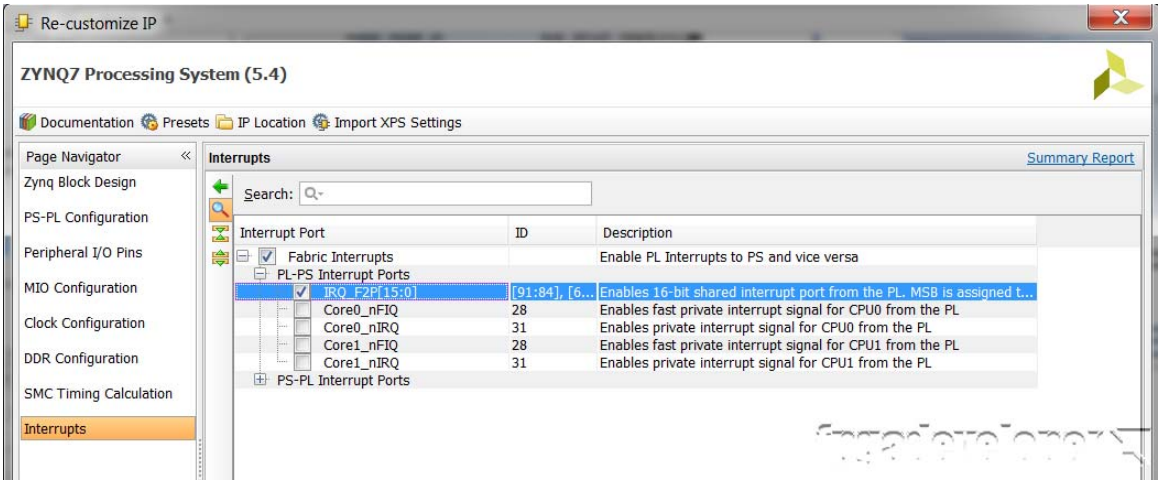
1. In the block diagram, double click the AXI DMA block.
2. Un-tick the 'Enable Control / Status Stream' option and click OK.



Connect the DMA interrupts to the PS

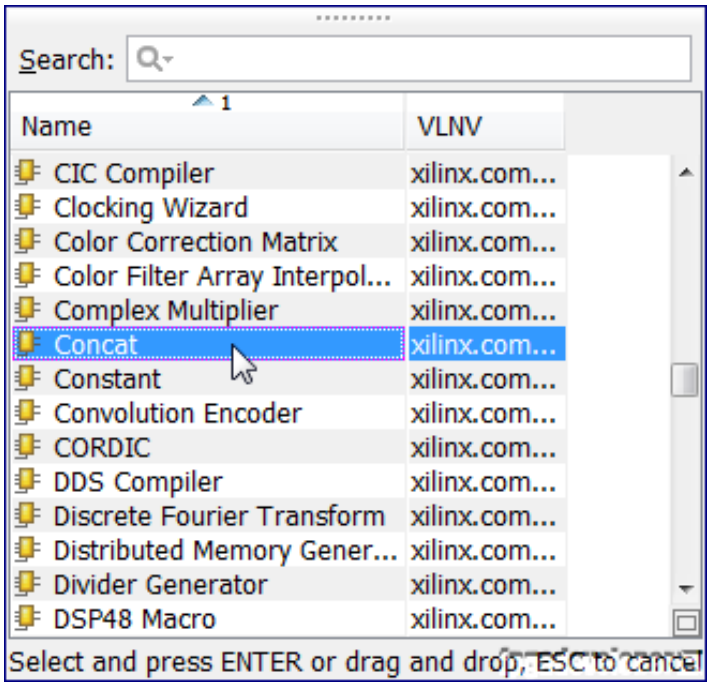
Our software application will test the DMA in polling mode, but to be able to use it in interrupt mode, we need to connect the interrupts 'mm2s_introut' and 's2mm_introut' to the Zynq PS.

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

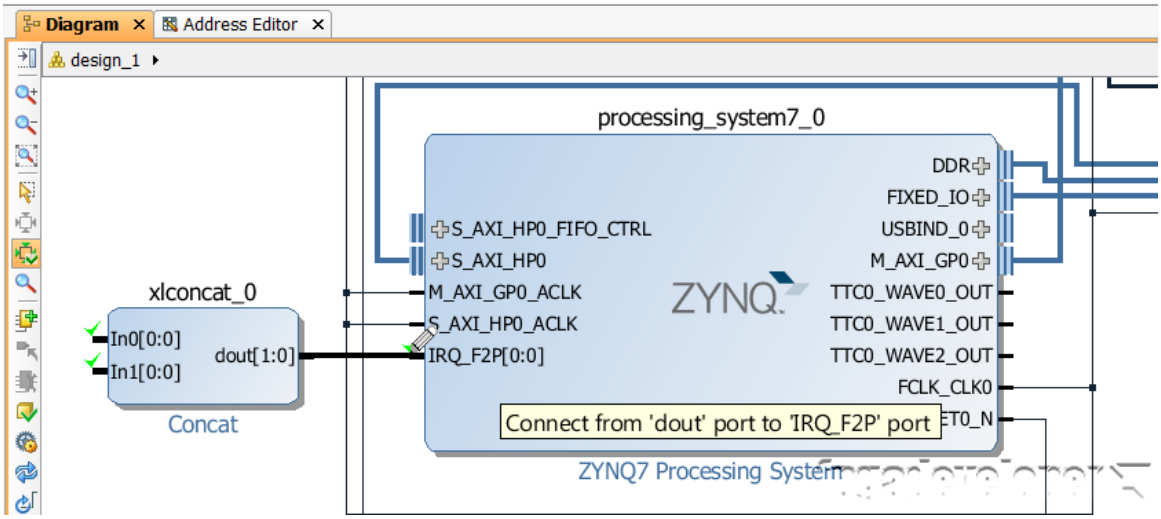


2. Tick 'Fabric Interrupts' and 'IRQ_F2P[15 :0]' to enable them, and click OK.

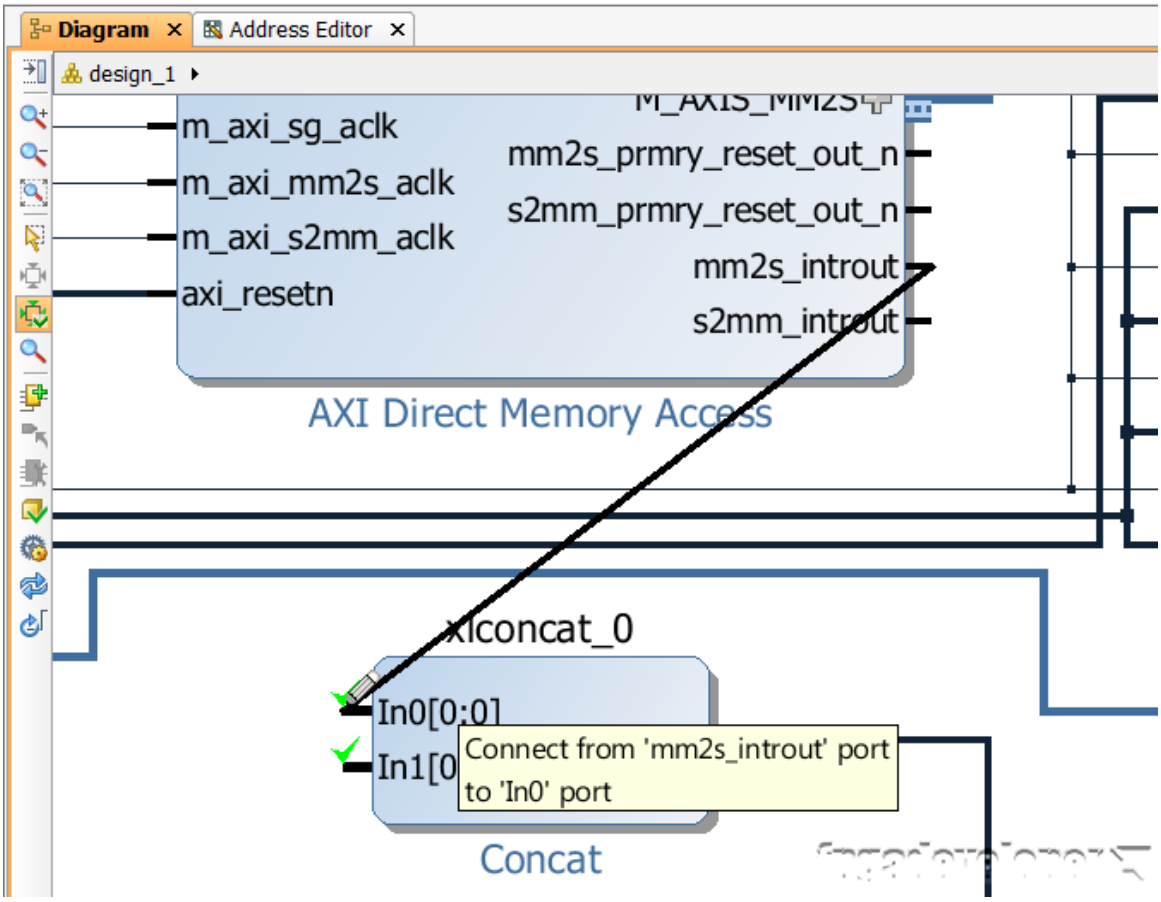
3. Click the 'Add IP' icon and double-click 'Concat' from the catalog.



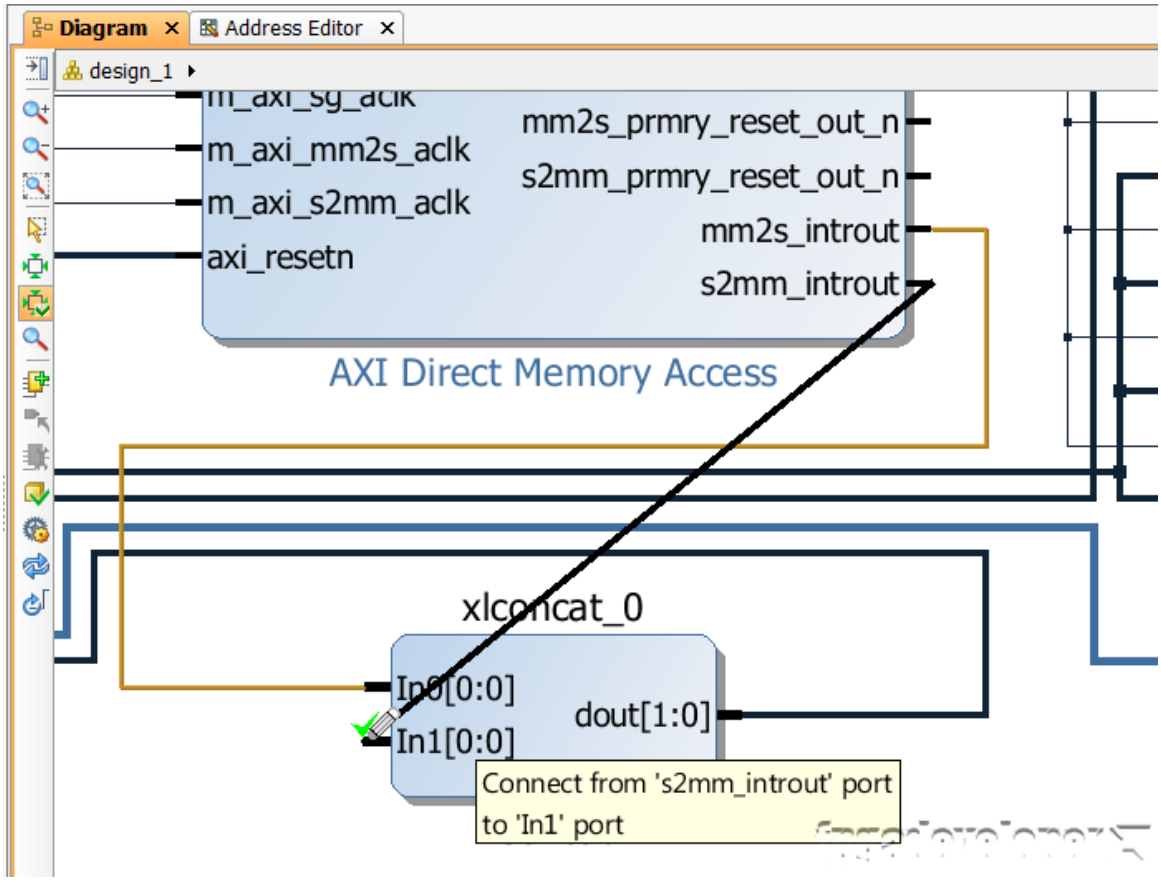
4. Connect the 'dout' port of the Concat to the 'IRQ_F2P' port of the Zynq PS.



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

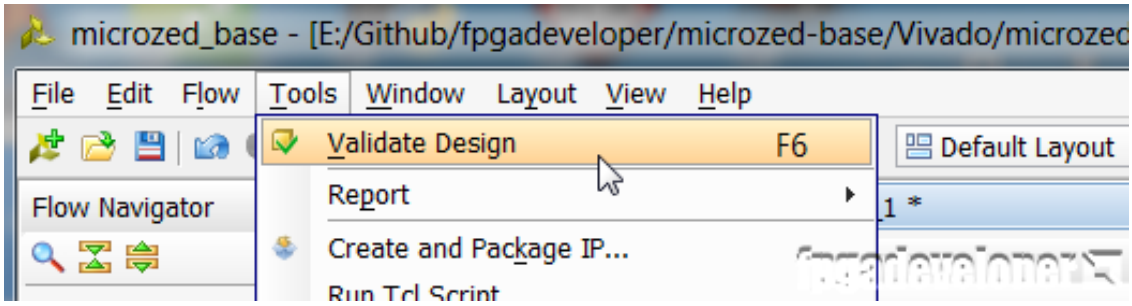


6. Connect the 's2mm_introut' port of the DMA to the 'In1' port of the Concat.



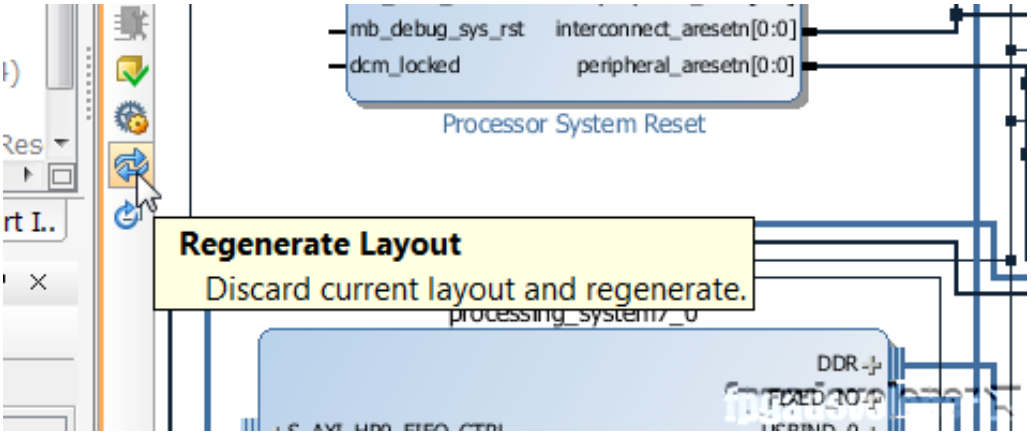
Validate and build the design

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

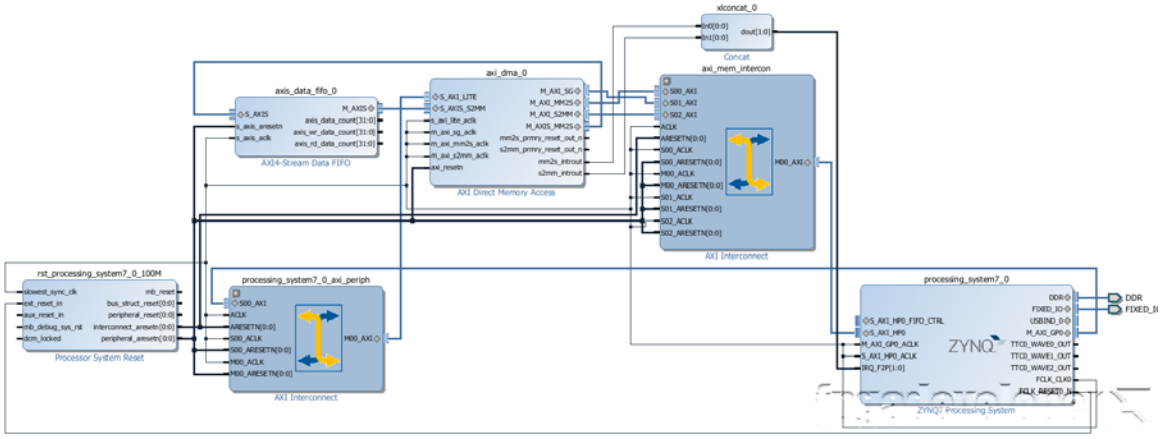


2. You should get this message saying that validation was successful.

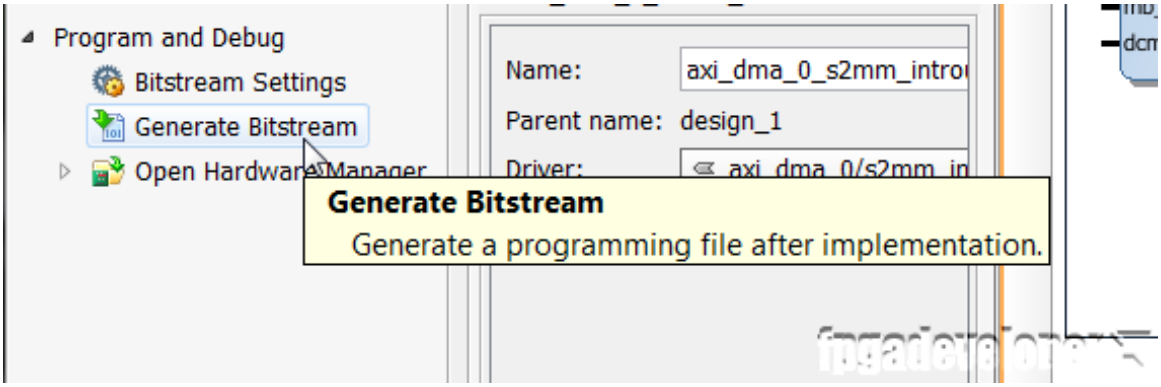
3. We can clean up the block diagram by clicking the Regenerate Layout icon.



4. Our block diagram now looks like this :



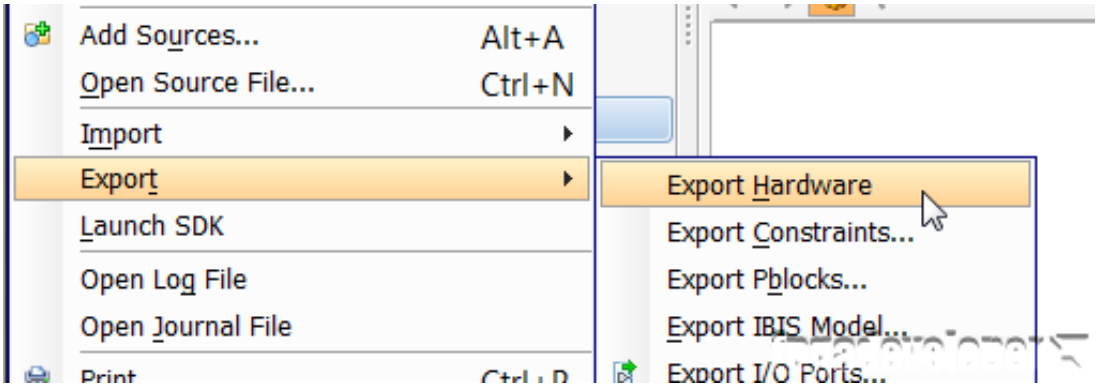
5. In the Flow Navigator, click ‘Generate Bitstream’.



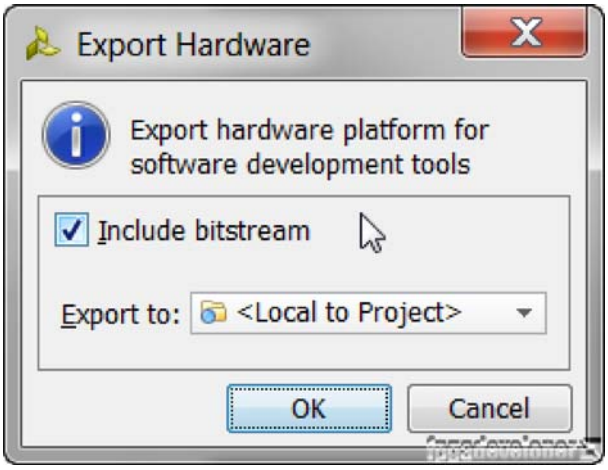
Export the hardware design to SDK

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.

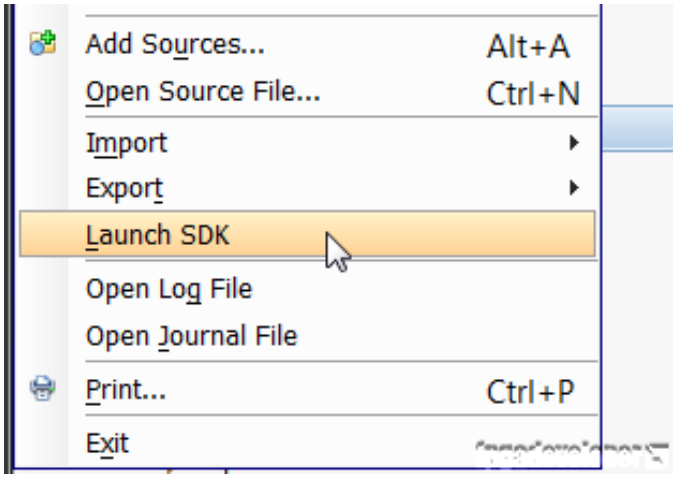
1. In Vivado, from the File menu, select “Export->Export hardware”.



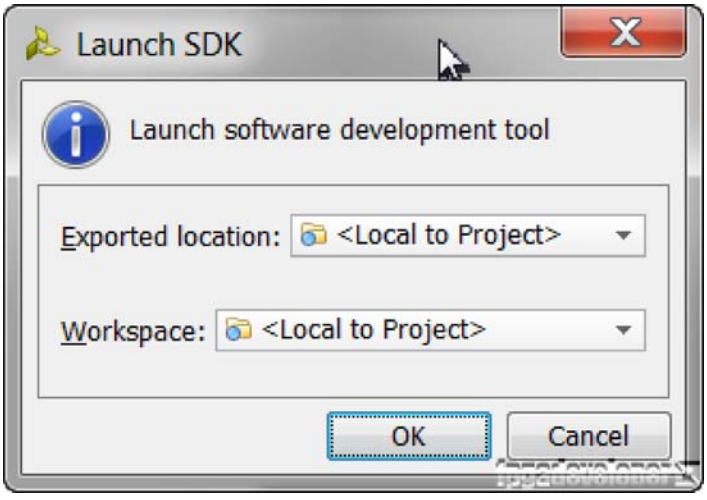
2. In the window that appears, tick “Include bitstream” and click “OK”.



3. Again from the File menu, select “Launch SDK”.

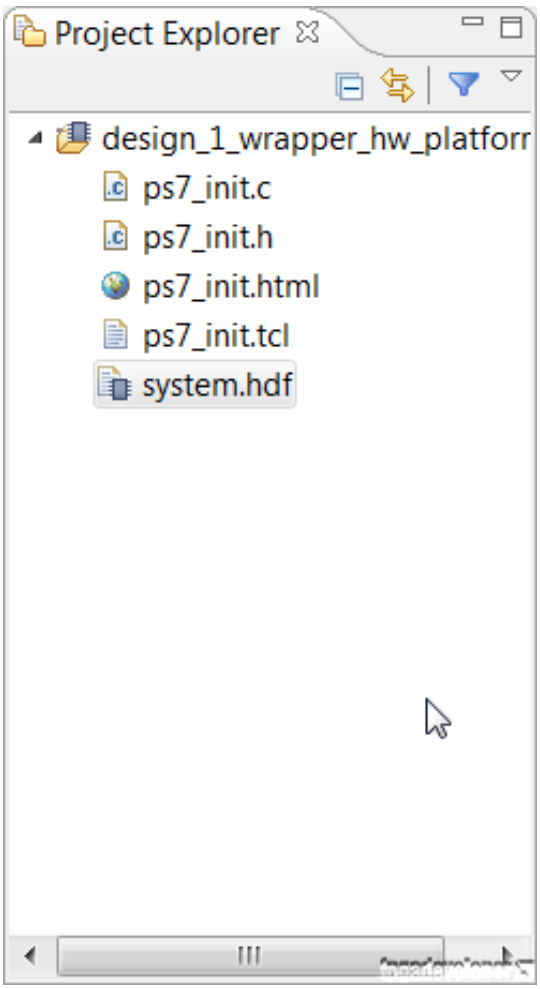


4. In the window that appears, use the following settings and click “OK”.



At this point, the SDK loads and a hardware platform specification will be created

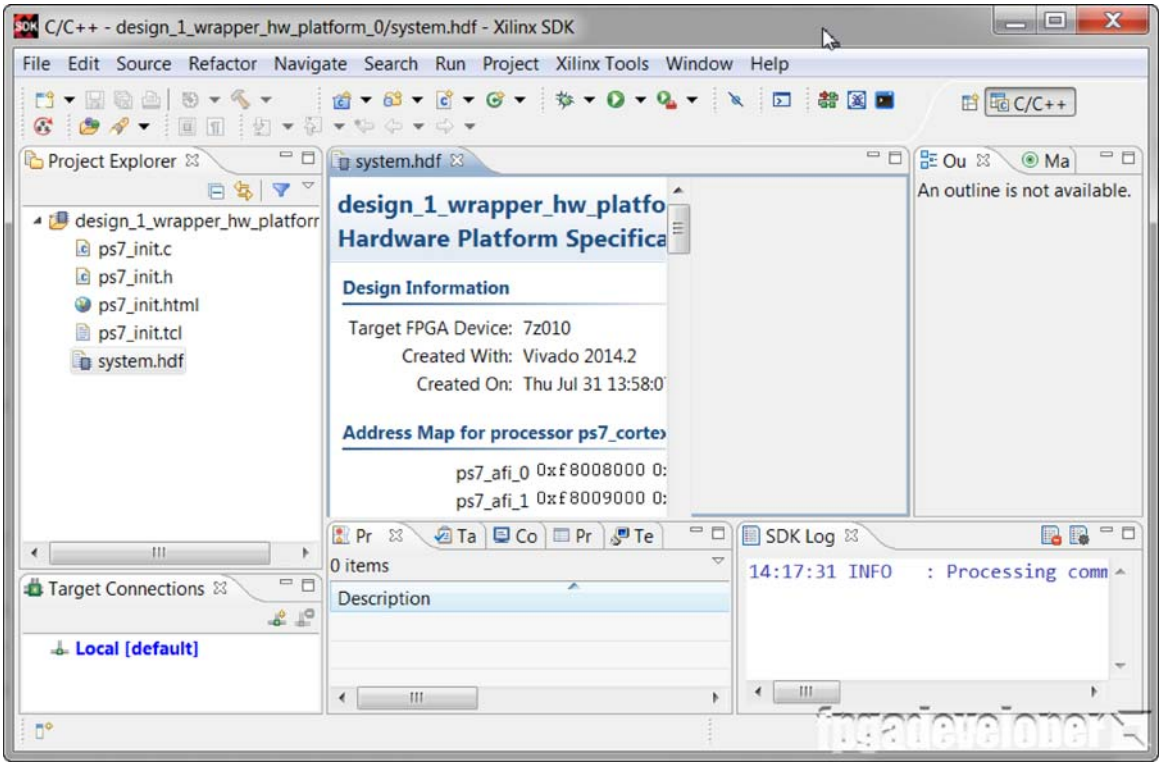
for your design. You should be able to see the hardware specification in the Project Explorer of SDK as shown in the image below.



We are now ready to create the software application.

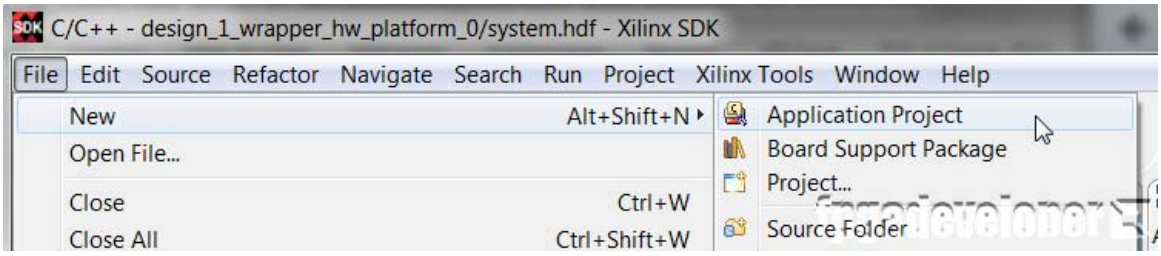
Create a Software application

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

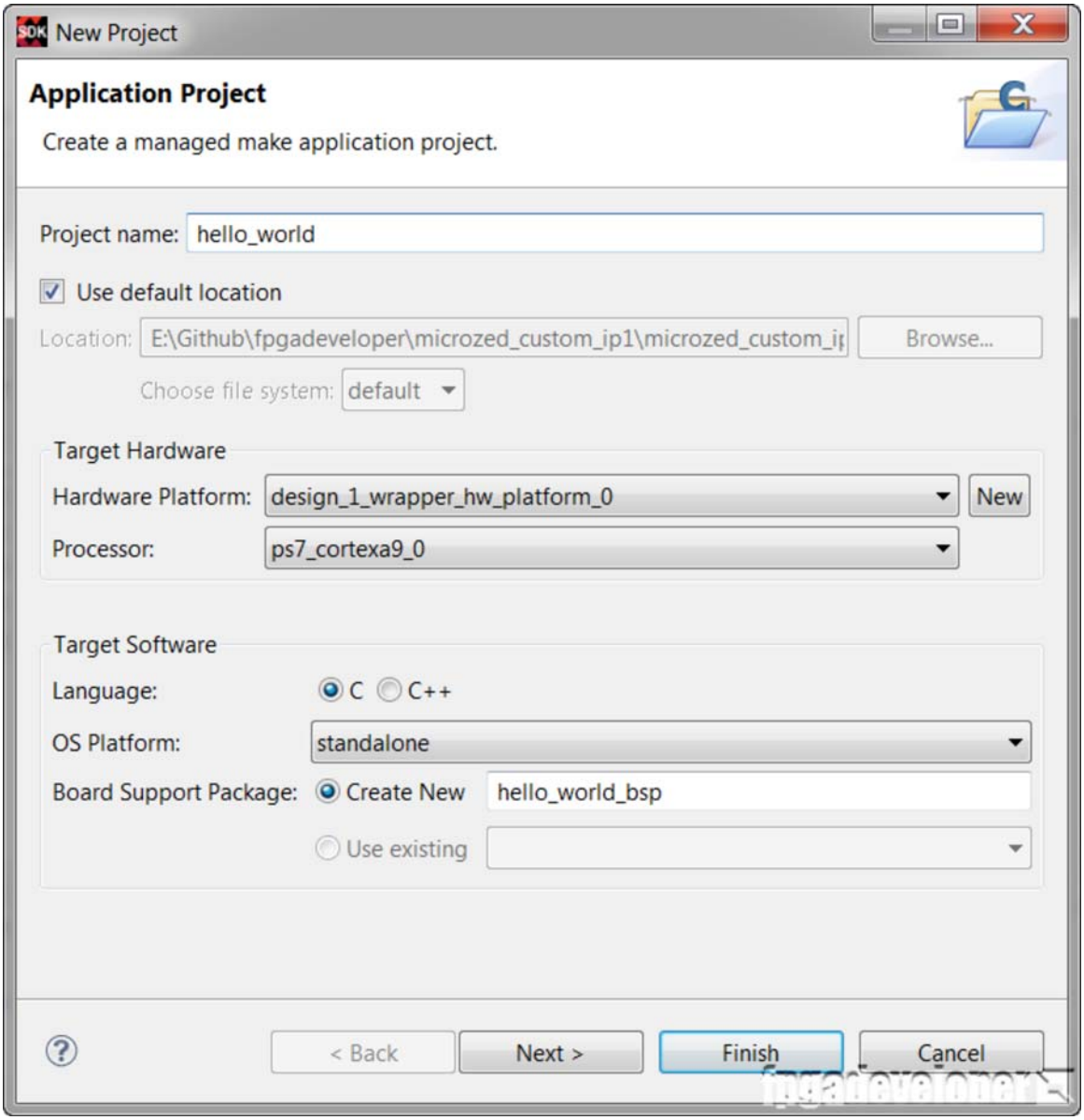


To make things easy for us, we'll use the template for the hello world application and then modify it to test the AXI DMA.

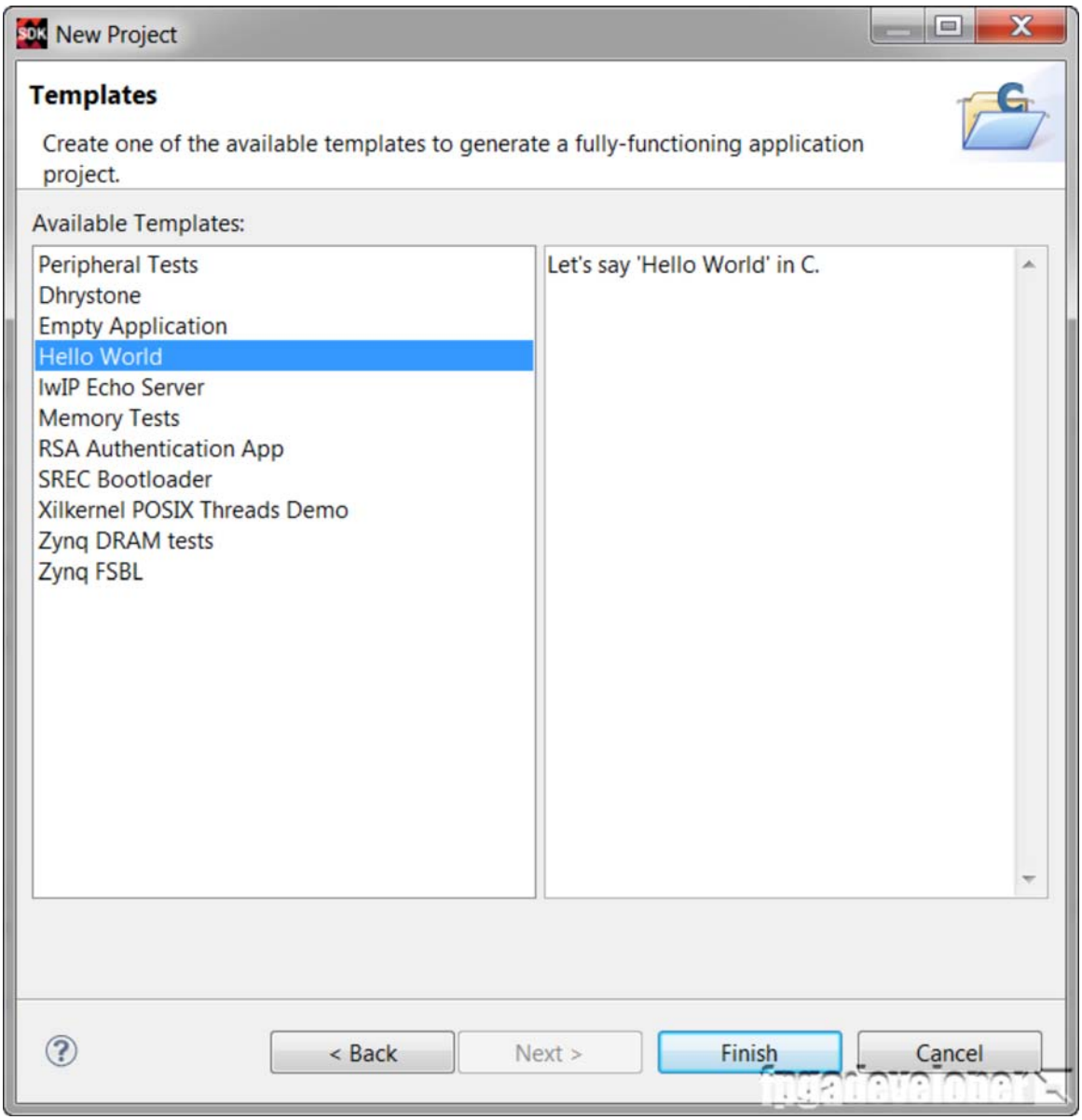
1. From the File menu, select New->Application Project.



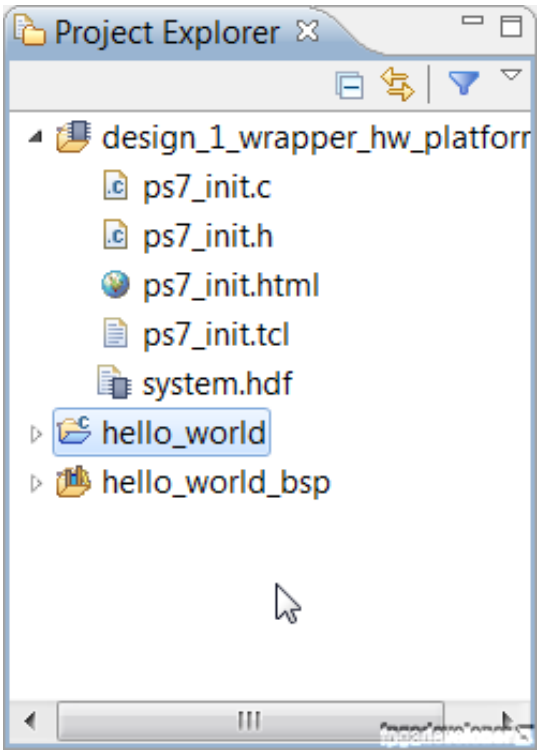
2. In the first page of the New Project wizard, choose a name for the application. I've chosen "hello_world". Click "Next".



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



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



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

Modify the Software Application

We need to modify the hello world software application to test our DMA.

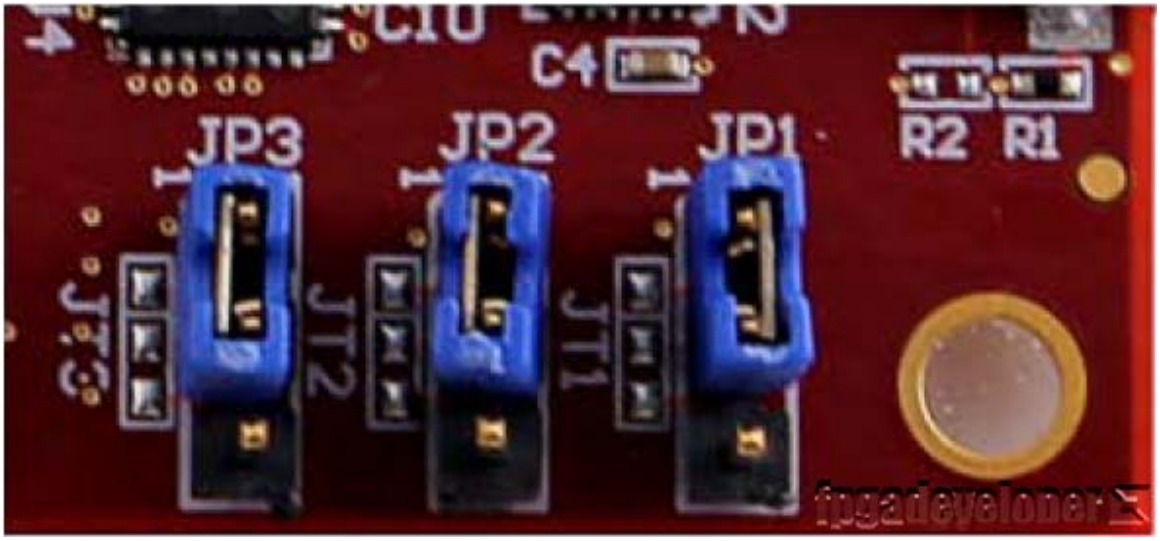
1. From the Project Explorer, open the “hello_world/src” folder. Open the “helloworld.c” source file.
2. Replace all the code in this file with the code that you will find on Github here: https://github.com/fpgadeveloper/microzed-axi-dma/blob/master/SDK/hello_world/src/helloworld.c
3. Save and close the file. The application should build automatically.

> The application source code is derived from an example provided by Xilinx in the installation files. You can find it at this location on your PC:
>
> `C:\Xilinx\14.7\ISE_DS\EDK\sw\XilinxProcessorIPLib\drivers\axidma_v7_02_a\examples\xaxidma_example_sg_poll.c`
>
> By the way, if you didn’t know about it already, that folder contains heaps of examples that you will find useful, I suggest you check it out.

Test the design on the hardware

To test the design, we are using the MicroZed board from Avnet. Make the following setup before continuing:

1. On the MicroZed, set the JP1, JP2 and JP3 jumpers all to the 1-2 position.

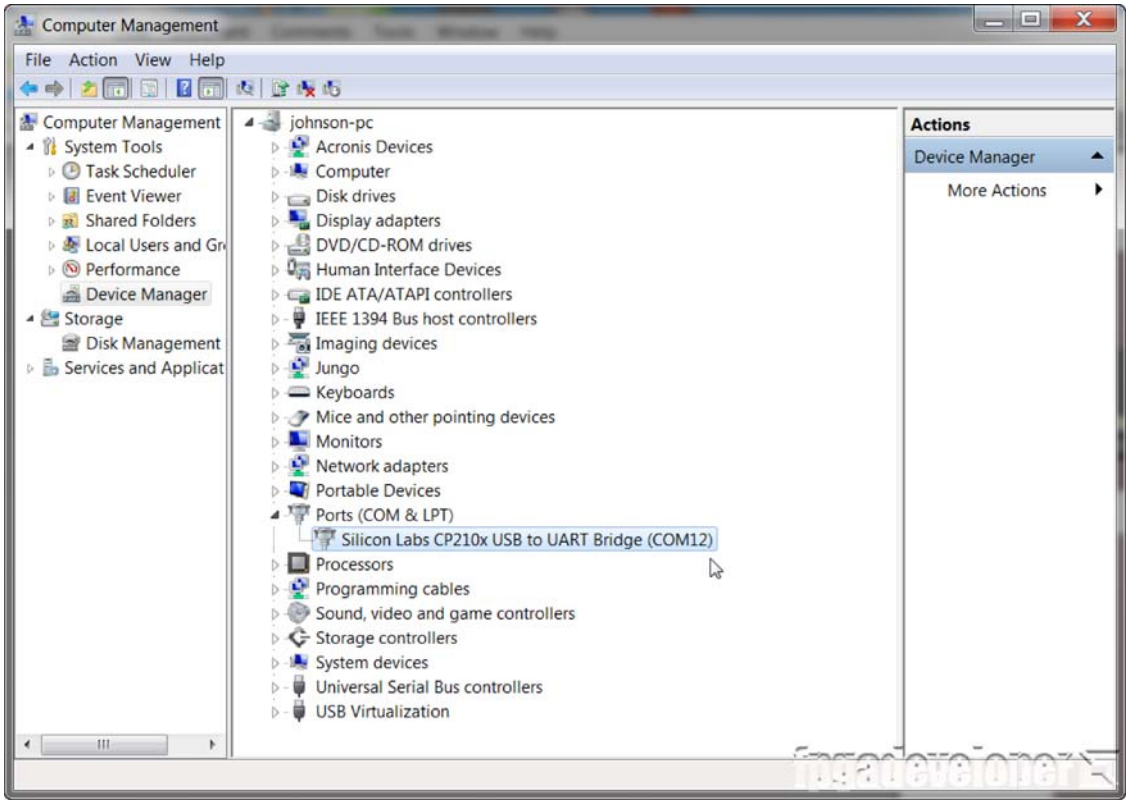


2. Connect the USB-UART (J2) to a USB port of your PC.

3. Connect a Platform Cable USB II programmer (or similar device) to the JTAG connector. Connect the programmer to a USB port of your PC.

Now you need to open up a terminal program on your PC and set it up to receive the test messages. I use Miniterm because I'm a Python fan, but you could use any other terminal program such as Putty. Use the following settings:

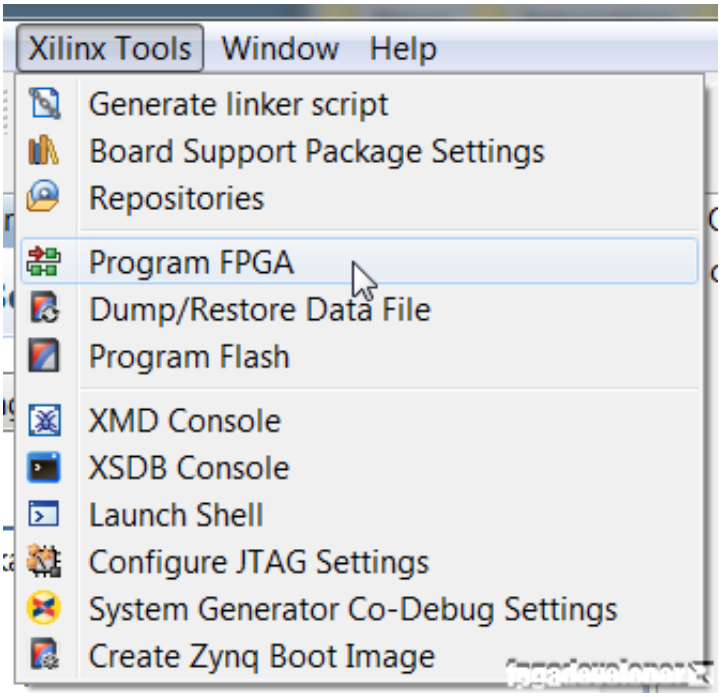
- Comport – check your device manager to find out what comport the MicroZed popped up as. In my case, it was COM12 as shown below.



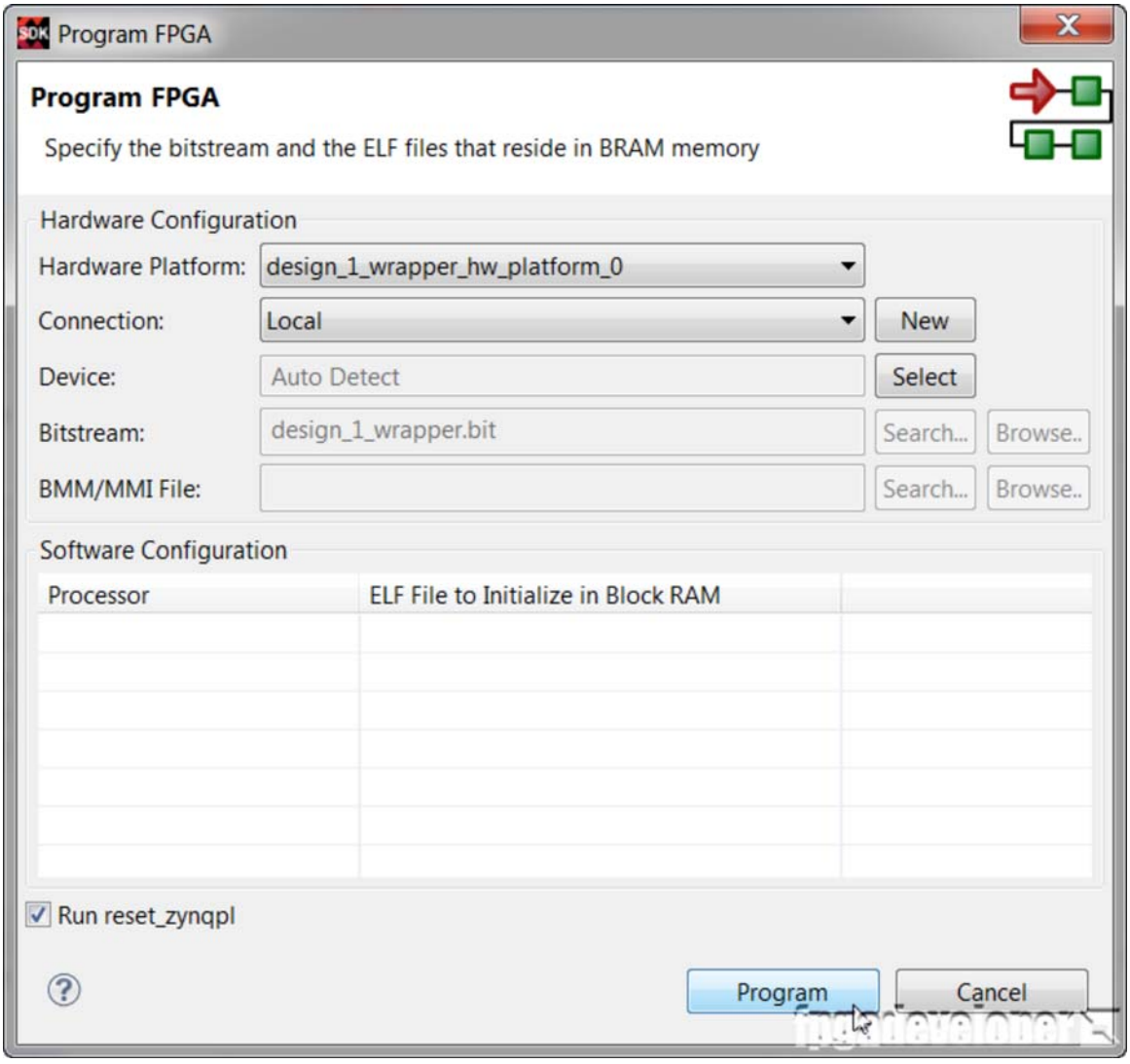
- Baud rate: 115200bps
- Data: 8 bits
- Parity: None
- Stop bits: 1

Now that your PC is ready to receive the test messages, we are ready to send our bitstream and software application to the hardware.

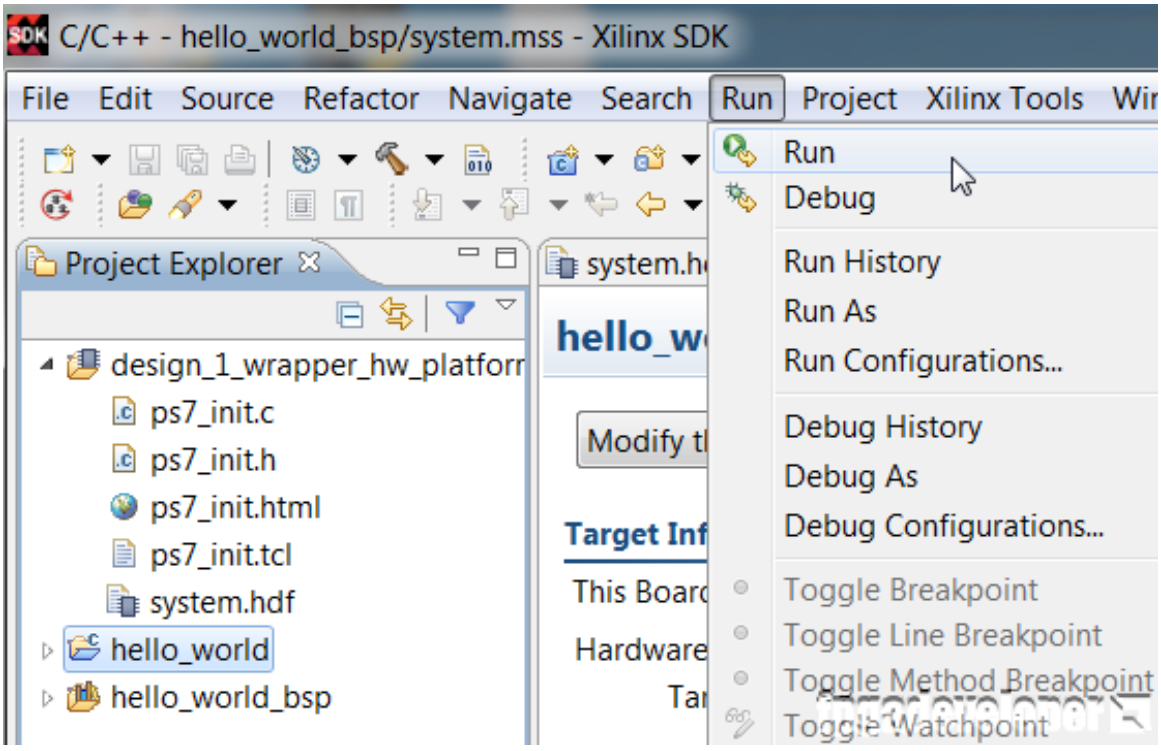
1. In the SDK, from the menu, select Xilinx Tools->Program FPGA.



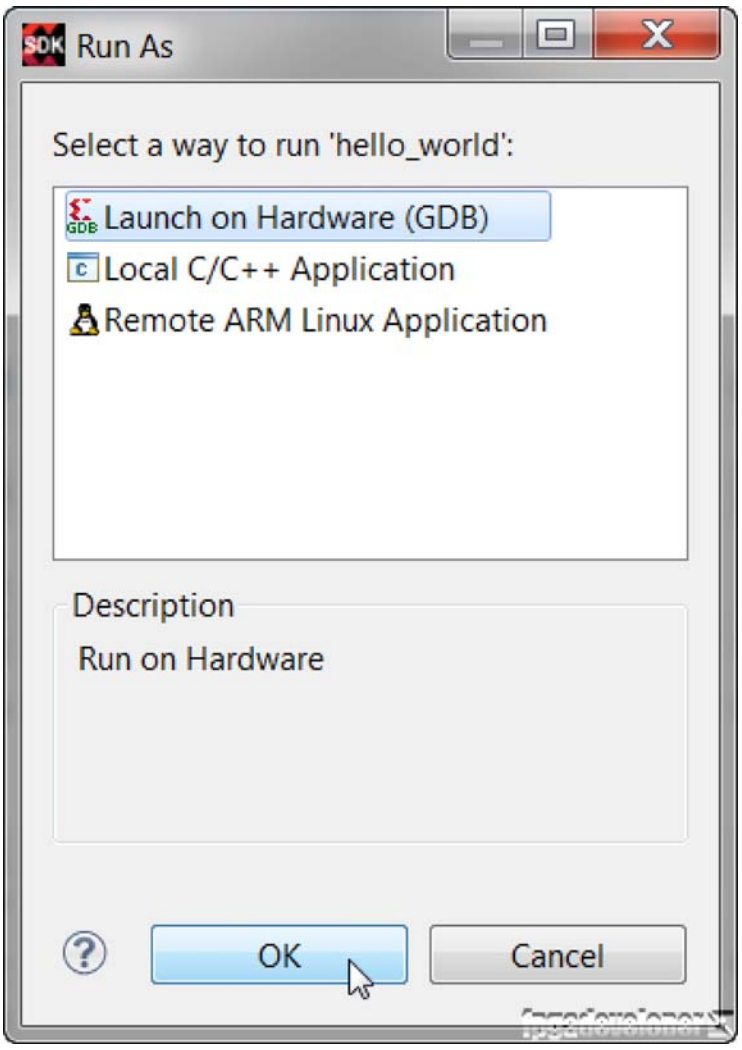
2. In the Program FPGA window, we select the hardware platform to program. We have only one hardware platform, so click “Program”.



3. The bitstream will be loaded onto the Zynq and we are ready to load the software application. Select the "hello_world" folder in the Project Explorer, then from the menu, select Run->Run.



4. In the Run As window, select "Launch on Hardware (GDB)" and click "OK".



5. The application will be loaded on the Zynq PS and it will be executed. Look out for the results in your terminal window!

Source code

The TCL build script and source code for this project is shared on Github at the following links:

- For the MicroZed: https://github.com/fpgadeveloper/microzed-axi-dma
- For the ZedBoard: https://github.com/fpgadeveloper/zedboard-axi-dma

For instructions on rebuilding the project from sources, read my post on version control for Vivado projects.

Jeff Johnson

Jeff holds a bachelors degree in Electrical Engineering from the University of Sydney, and has more than a decade of experience in electronic and FPGA design. He has worked for design houses in Australia and Canada developing electronic products for a wide range of industries and markets. Jeff now works as an electronic design consultant and offers electronic and FPGA design services through his company Opsero.

19 Comments

juncai on August 14, 2014 at 1:46 am

3 0 Rate This

Thanks for your tutorial, it help me lot.

Do you have an axidma application example on linux? I'm more interested in linux running than bere-metal application.

thanks in advance!

Reply

Paul Thomas on August 17, 2015 at 7:52 am

0 0 Rate This

juncai, you can the xilinx dmaengine driver for this example. Xilinx, even has test kernel module, axidmatest. It hasn't made it to the mainline yet, so you have to use their github branch here: https://github.com/Xilinx/linux-xlnx/tree/master/drivers/dma/xilinx

thanks,
Paul

Reply

M on August 14, 2014 at 4:14 pm

0 0 Rate This

Thanks for this tutorial! Are there any changes that need to be made for the helloworld.c code (the one that you shared) to run on a ZedBoard?

Reply

kA on August 27, 2014 at 2:41 am

0 0 Rate This

I can answer this one — it works just fine as-is on the regular Zedboard.

Reply

Amir on August 15, 2014 at 11:25 am

0 0 Rate This

Hi Jeff, thanks for this great tutorial. May I ask, how about MicroBlaze processor? Is there anything should I change/modify?

Kind regards,
Amir.

Reply

F on August 25, 2014 at 10:32 am

0 0 Rate This

Thank you for the tutorial! I´ve been experimenting lately with the DMA in Simple Mode (No Scatter Gather). In this example as well it seems that RX_BUFFER_BASE has to be 32 BYTES (not 32 bit) alligned. Is this supposed to be the expected behaviour? I think it has something to do with the "Xil_DCacheInvalidateRange" function.

Greetings,
Fran

Reply

Hendrik on September 2, 2014 at 9:24 am

0 0 Rate This

Hi,

in my case the provided example works well, but I also experience difficulties in the Simple Mode. I tried different things (with the

software based on the examples provided by Xilinx): with/without interrupts, with/without Micro DMA, with/without unaligned adresses..
However, none of them works. Did you get it finally running in some way?

Best regards,
Hendrik

Reply

Adi on September 8, 2014 at 3:37 am

0 0 Rate This

Hi Jeff, thanks for the example. I'm a little confused by the fact that there is only a HP Slave connection to the PS7 for the DMA data. Is the DMA module effectivly reading data from the PS7 and writing it to the FIFO. Then when the data loops back from the FIFO the DMA engine writes that back the PS7?

Reply

Fran on September 9, 2014 at 4:36 am

0 0 Rate This

Yes, I got it running connected to a HLS generated IP (very simple IP just for the test). If you are interested contact me at francisco-de-asis.molina-martel@iosb.fraunhofer.de.

Regards,
Fran

Reply

Girish Halady on September 13, 2014 at 3:54 pm

2 0 Rate This

Thanks for this AXI DMA Example, would it be possible to create a similar example DMA to a custom IP on Linux using DMA Device drivers to a custom memory IP added in the PL as a PS memory extension?

Thanks
Girish

Reply

Alex on October 26, 2014 at 1:23 am

0 0 Rate This

Can you post something related to XIP (execute in place) for zedboard in steps. It's hard concept of understanding.
Xilinx provided example is little bit confusing. Here the link:
http://www.wiki.xilinx.com/Zynq-7000+AP+SoC+Boot+-+Booting+and+Running+Without+External+Memory+Tech+Tip

Reply

Eldar on December 12, 2014 at 1:07 am

1 0 Rate This

Hi. Thanks for your tutorial. It seems vary helpfull for beginners.

Reply

Nabeel Anjum on December 16, 2014 at 8:51 pm

1 0 Rate This

hello !
It is indeed useful tutorial ! i need to ask one question ! how can we write our own custom VHDL code in it so that it can be interfaced with memory locations from PS !!

Reply

Jose Luis on January 11, 2015 at 6:06 pm

0 0 Rate This

Same question :)

Reply

Jose Luis on January 11, 2015 at 6:34 pm

0 0 Rate This

Hi.
In other of your post's (http://www.fpgadeveloper.com/2014/08/creating-a-custom-ip-block-in-vivado.html) you published the way to add custom vhdl code that get data from the axi-lite interface. Any source you can point to us to do something similar with the AXI DMA?
Thanks in advance and thanks for your incredible work with this post's.

Reply

Nabeel on April 2, 2015 at 9:38 am

0 0 Rate This

Dear,

I need to ask one question. Will this design work after disabling “Scatter Gather mode” option ? As in this design , M_AXI_SG is connecting to AXI interconnect only. I think by Simple DMA operation, it will work without changing any thing in Vivado except to disable SG option. Kindly tell if i am wrong : Thanks .

Reply

shrey on April 13, 2015 at 4:58 pm

0 0 Rate This

Hi i am using Zedboard and I want to use AXI DMA to transfer data from DDR to a block ram. I have tried using CDMA but it works faithfully only if I store my data on some parts of my DDR(that is not enough space for my application). I want to try using AXI DMA so I found your design and will modify it. But can you tell me is there any specific region(on 512 MB DDR) where I can store my data(32 bit width and 61439 depth) and transfer that data to Block ram by configuring DMA?

Reply

Vibha on June 9, 2015 at 11:40 pm

0 0 Rate This

Hi Jeff,
Thanks for the tutorial. Can you please suggest the changes required in hello world.c for testing this example on zc706.

Thanks in anticipation

Reply

Ning on August 14, 2015 at 4:15 am

0 0 Rate This

Hi, thank you for this useful tutorial. How can I insert a user Peripheral used AXI-Stream to connect my DAC ?

Reply

Trackbacks/Pingbacks

1. Using the AXI DMA Engine | FPGA Developer - [...] Update 2014-08-06: This tutorial is now available for Vivado – Using the AXI DMA in Vivado [...]

Most popular posts

Posts

All | Today | This Week | This Month

Creating a Base System for the Zynq in Vivado

5/5 (5 votes)

Creating a project using the Base System Builder

5/5 (3 votes)

Use iMPACT to Download a Bit File

5/5 (2 votes)

List and comparison of FPGA companies

5/5 (2 votes)

How to read an NGC netlist file

5/5 (2 votes)

Recent posts

KickStarter Campaign Launched: OnCourse Goggles

Ethernet gets robust

Back in black

Using Chipscope and SDK at the same time

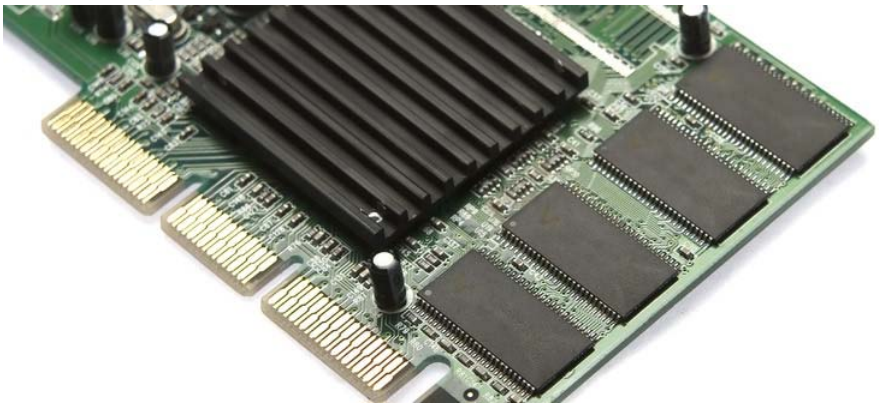
Sneak look at the new Robust Ethernet FMC

Topics

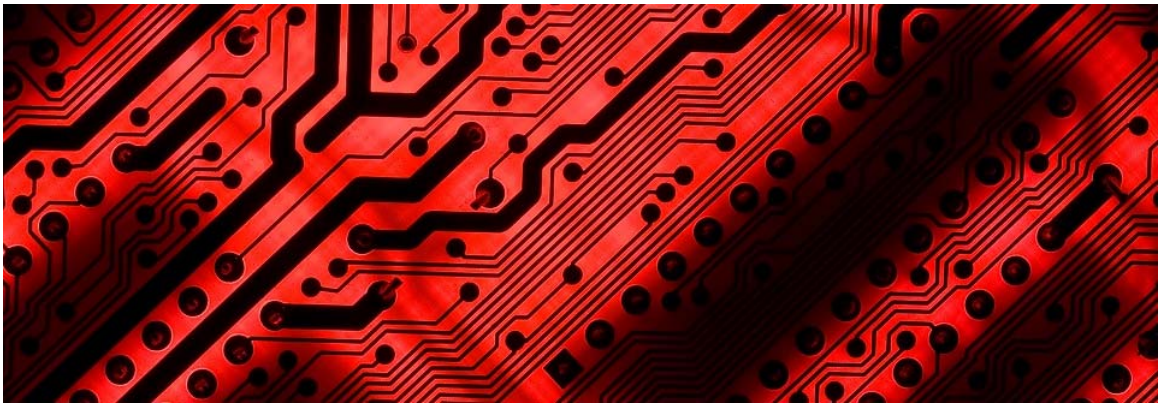
AC701 Aurora bsp chipscope custom ip dma Ethernet ethernet fmc finance FMC github hardware acceleration high frequency trading impact jtag KC705 MicroZed ML505/XUPV5 ML605 multigigabit transceiver ncd PCIe peripheral picozed rocketio sdk soc som svn tutorial VC707 VC709 version control Virtex-5 Virtex-6 Virtex-II Pro vivado xilinx XUPV2P ZC702 ZC706 ZedBoard ZYBO Zynq

Go to my company website for

FPGA design services



PCB design services



About Me

I'm an FPGA designer and I provide FPGA and PCB design services to innovative companies around the world. I believe in sharing knowledge and I regularly contribute to the open source community.

Contact me for FPGA design services

Recent Posts

- KickStarter Campaign Launched: OnCourse Goggles
- Ethernet gets robust
- Back in black
- Using Chipscope and SDK at the same time
- Sneak look at the new Robust Ethernet FMC

Topics

AC701 Aurora bsp chipscope custom ip dma Ethernet ethernet fmc finance FMC github hardware acceleration high frequency trading impact jtag KC705 MicroZed ML505/XUPV5 ML605 multigigabit transceiver ncd PCIe peripheral picozed rocketio sdk soc som svn tutorial VC707 VC709 version control Virtex-5 Virtex-6 Virtex-II Pro vivado xilinx XUPV2P ZC702 ZC706 ZedBoard ZYBO Zynq

Useful Links

- Digilent Inc
- EDA Board
- Ethernet FMC
- FPGA Consulting & Design Services
- fpga4fun

- Linux Wiki
- Opencores
- PCB Design Services for FPGA
- Xilinx Forum
- Xilinx Website