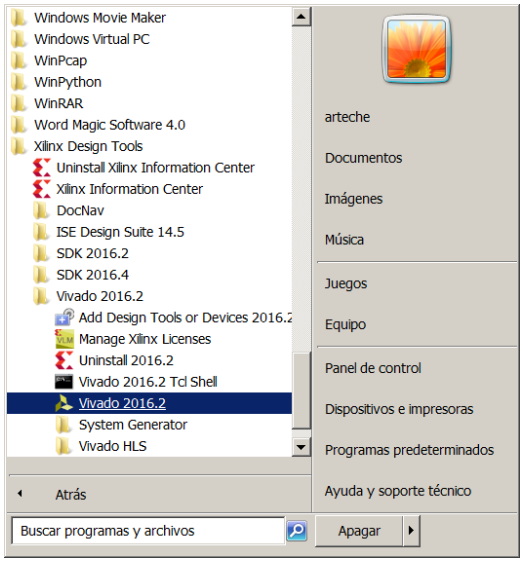
CREATING A BASE ZYNQ DESIGN WITH VIVADO 2016.2

based on:

http://www.zedboard.org/zh-hant/node/1454

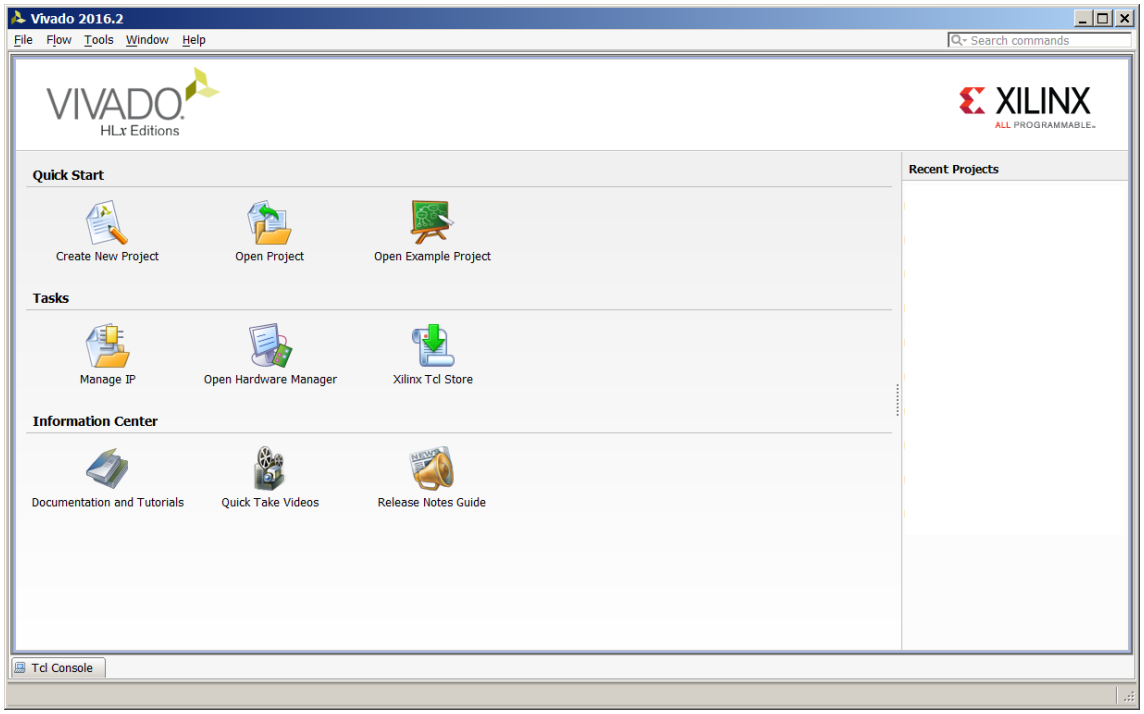
First, make sure you have Vivado + SDK 2016.2 installed (follow the how-to here.)

Next, launch Vivado 2016.2:



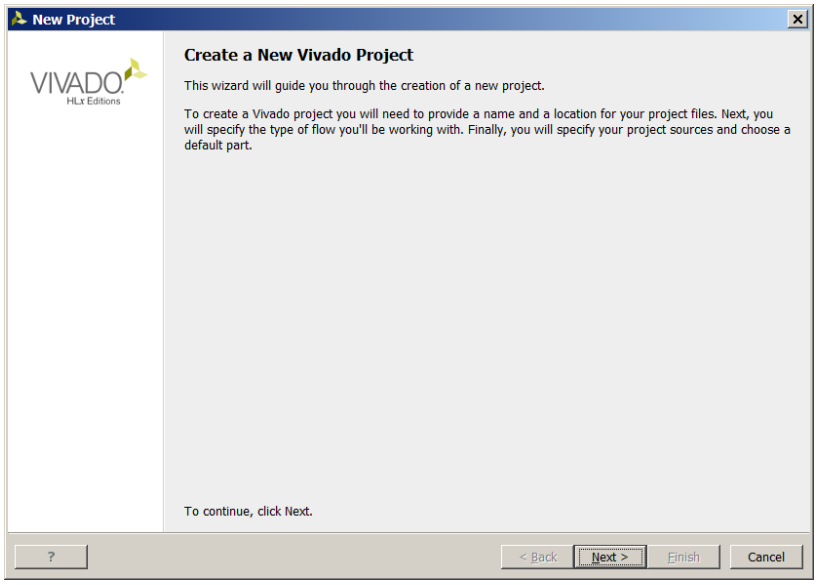
Launch Vivado 2016.2 from the start menú

Next we are presented with the launch screen where we have eight different options to choose from.



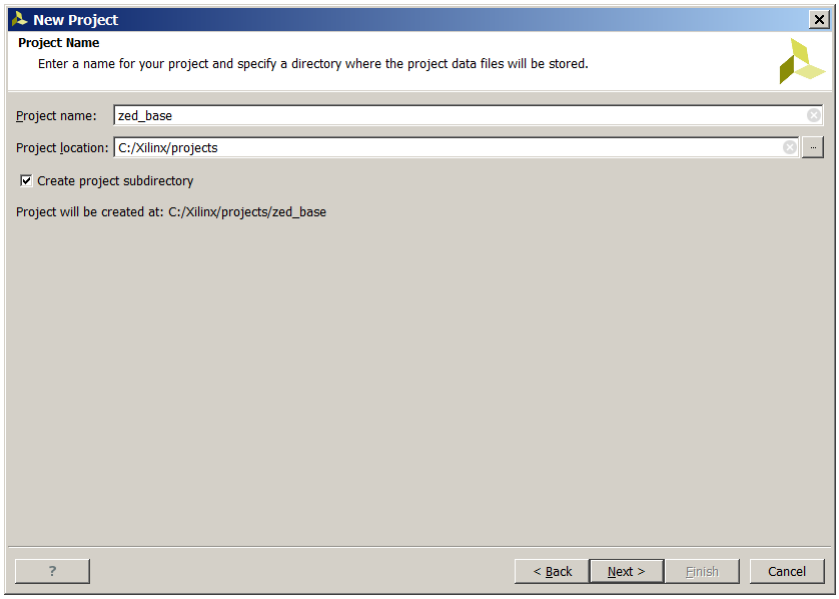
Vivado 2016.2 Launch Window.

From the launch window we have a number of different options. We can create a new project, open an existing project, open an example project, manage IP, view documentation and tutorials, view user guides for Xilinx silicon and tools, view informational videos about Vivado and the Xilinx 7-Series silicon devices, as well as view release notes for the current version of Vivado. Click Create New Project to continue.



The 'Create a New Vivado Project' wizard is launched.

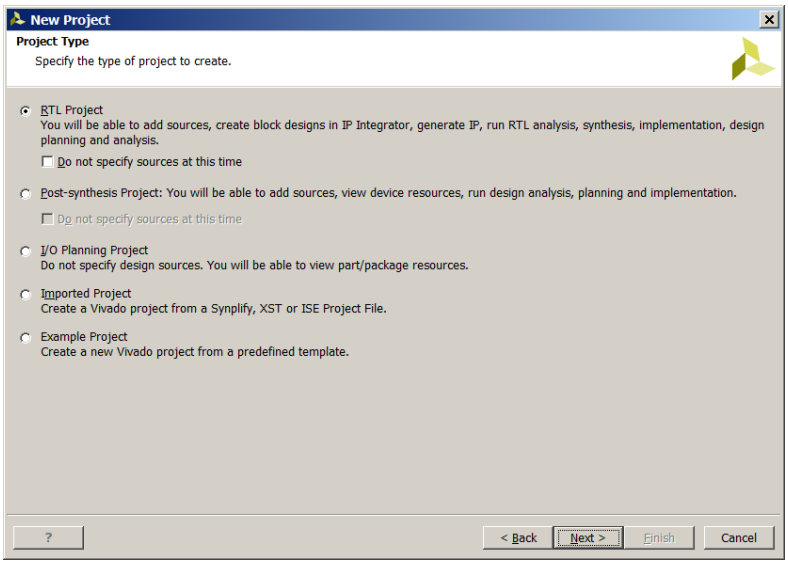
The 'Create a New Vivado Project' wizard will help us create a new project targeting a specific part (and/or development kit). It also will allow us to add existing design files, such as HDL that you have previously written on other projects, as well as constraint files such as pin outs and timing constraints, and finally packaged IP. Click **Next** to continue..



The first 'real' page of the new project wizard.

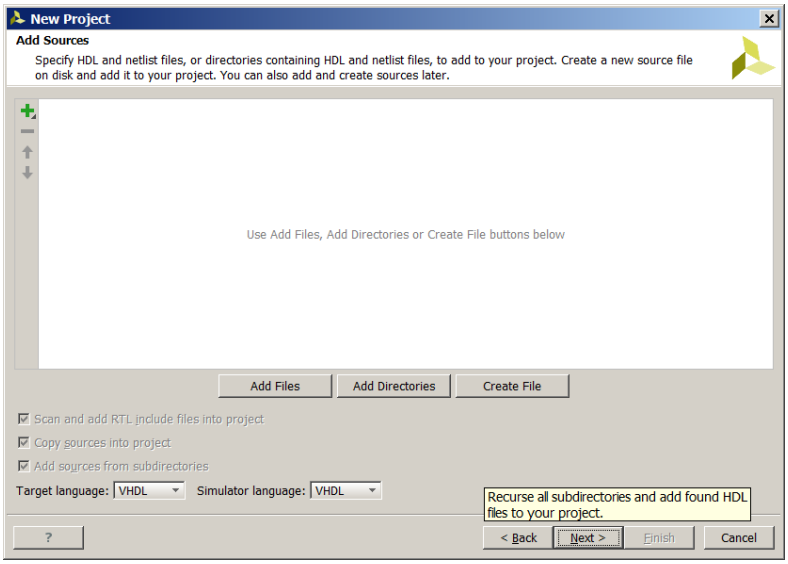
You are going to want to name your project on this page of the wizard. You also have the ability to place the project within a specific directory.I would recommend keeping directory paths short

and never, EVER, using spaces in your file/folder names. Name your project and click Next (you may want to keep the same name for this first run through to make things as simple as possible).



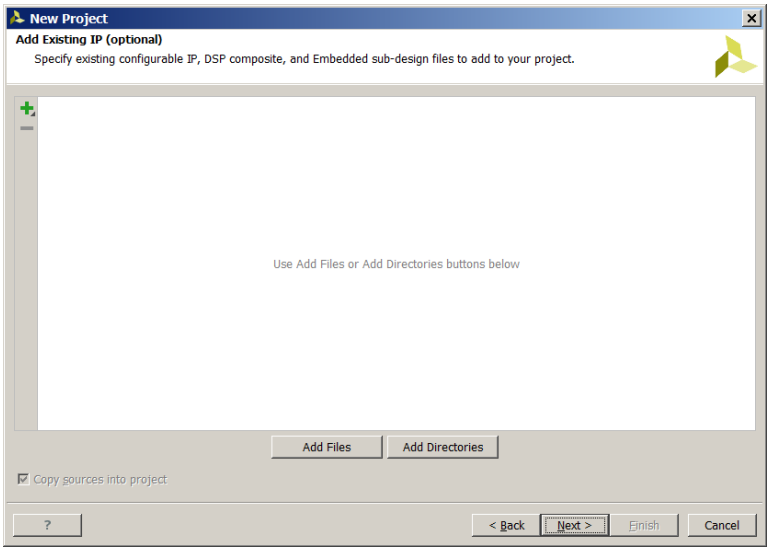
Project type selection page.

There are a few different types of project that Vivado supports. We will be creating a RTL project for your base Zynq design. Select 'RTL Project' and click Next. Note: I did not check 'Do not specify sources at this time' so we can see all of the pages of the wizard, but for new projects I traditionally do check this and add the sources later within the Vivado GUI.



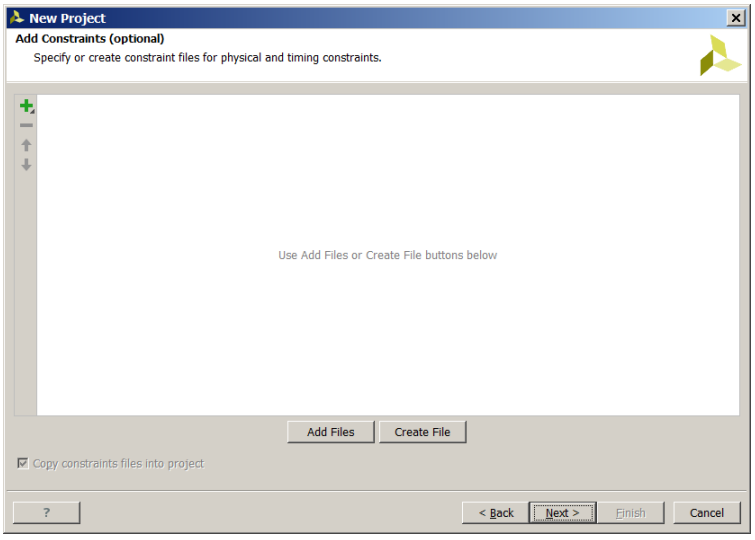
Page to add sources to the project and select the default language.

Like in the previous step, we will not be adding any sources to this design (we don't have anything to add!). If you prefer VHDL you have to change the default Target Language from Verilog to VHDL. Click Next to continue.



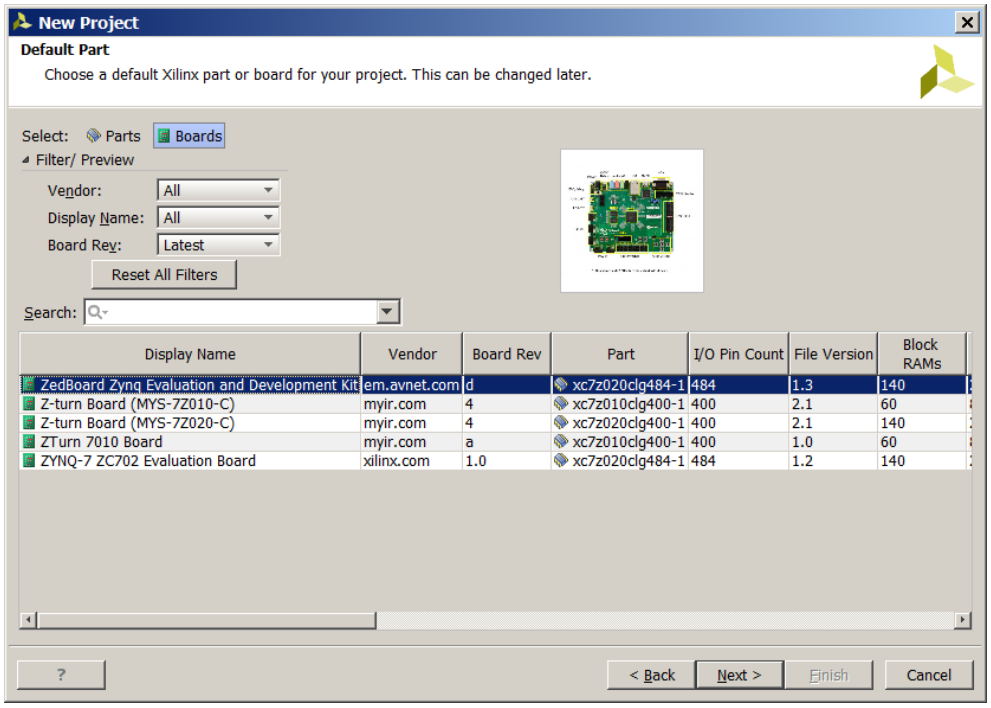
Add existing IP to your design.

Vivado has the ability to package designs up into portable IP packages. For now, just click Next.



Add constraints to your design.

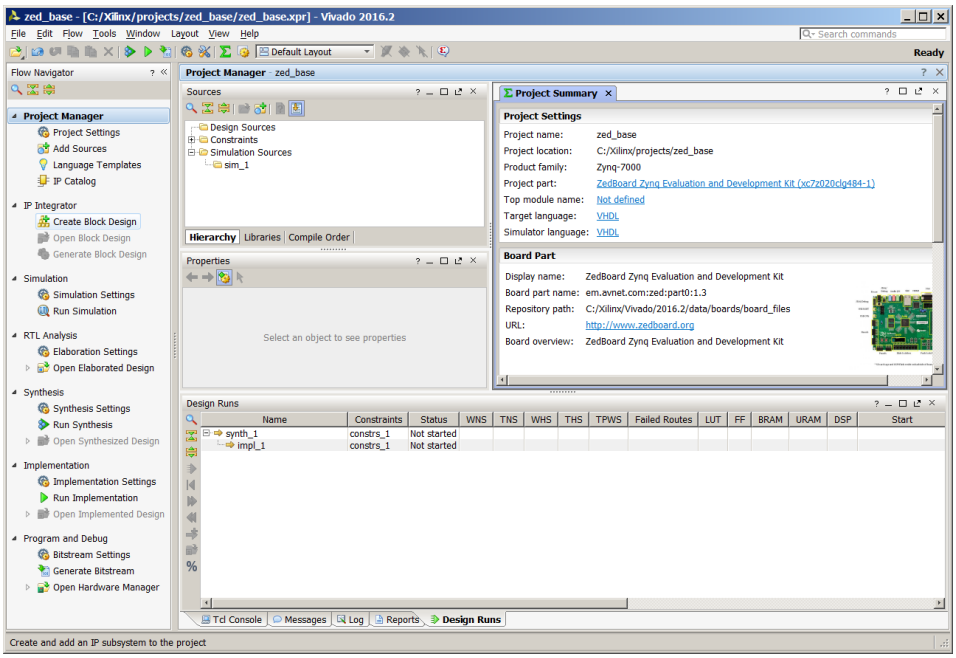
Vivado uses the XDC format, which is a series of TCL commands. We will not be adding any constraints to this design, but if we needed too, we would be doing so here. Click Next to continue.



Select the device you will be targeting.

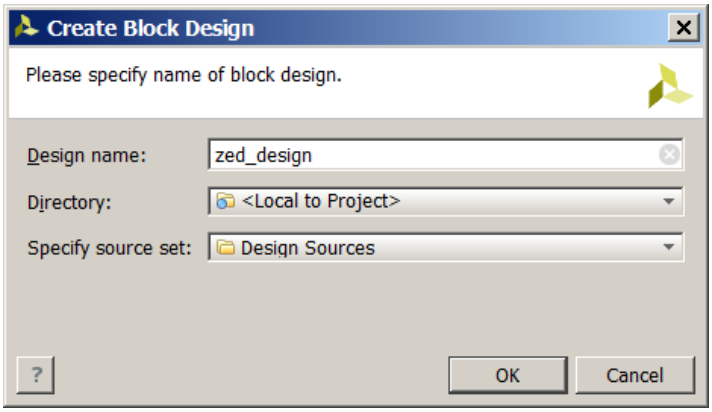
This page is very important. We will be selecting which 7-Series device we will be targeting for our Vivado project. There are two flows that can be seen in the upper left of the page: Parts and Boards. I will be basing this tutorial on the Zedboard, so I am going to go with the Boards flow. If you are using custom hardware, or a third part board you will need to select your part via the Parts flow. Click 'Boards' in the top left, and then select the Zedboard development kit from the list. Click Next to continue.

The last page of the wizard is a summary of your selection. Review it, and click Finish



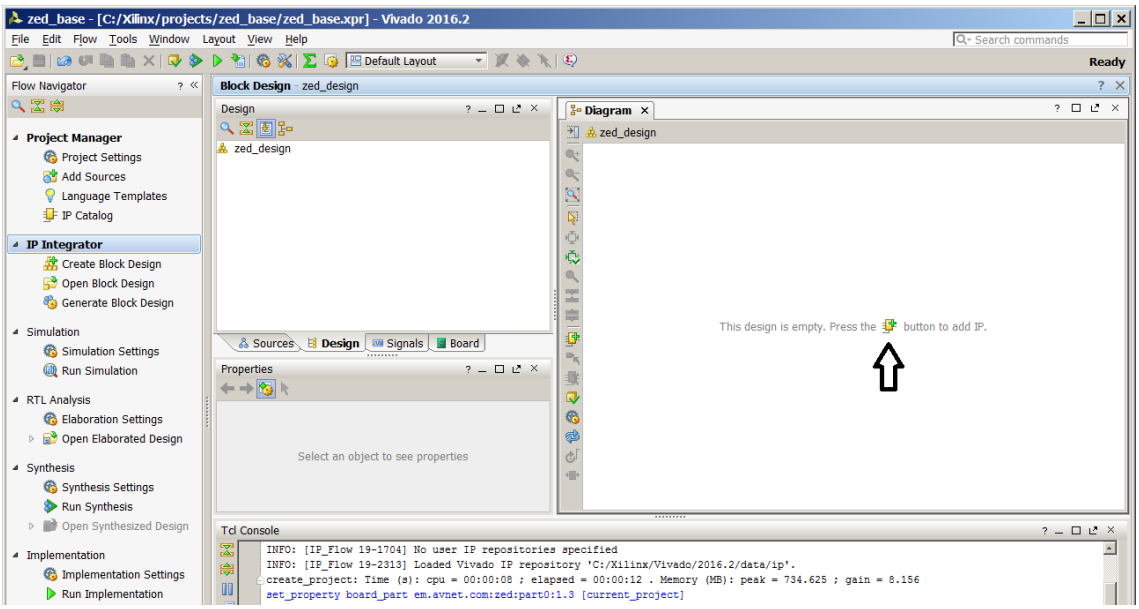
Default view of Vivado post new project wizard.

Now that you have configured your project with the new project wizard, you now are presented with the default view of Vivado. You will see the Flow Navigator on the left side of the window. The Flow Navigator is where you will be launching the various steps of your design process, including creating a new Block Design with IPI. Find the IP Integrator tree item, expand it, and select 'Create Block Design'.



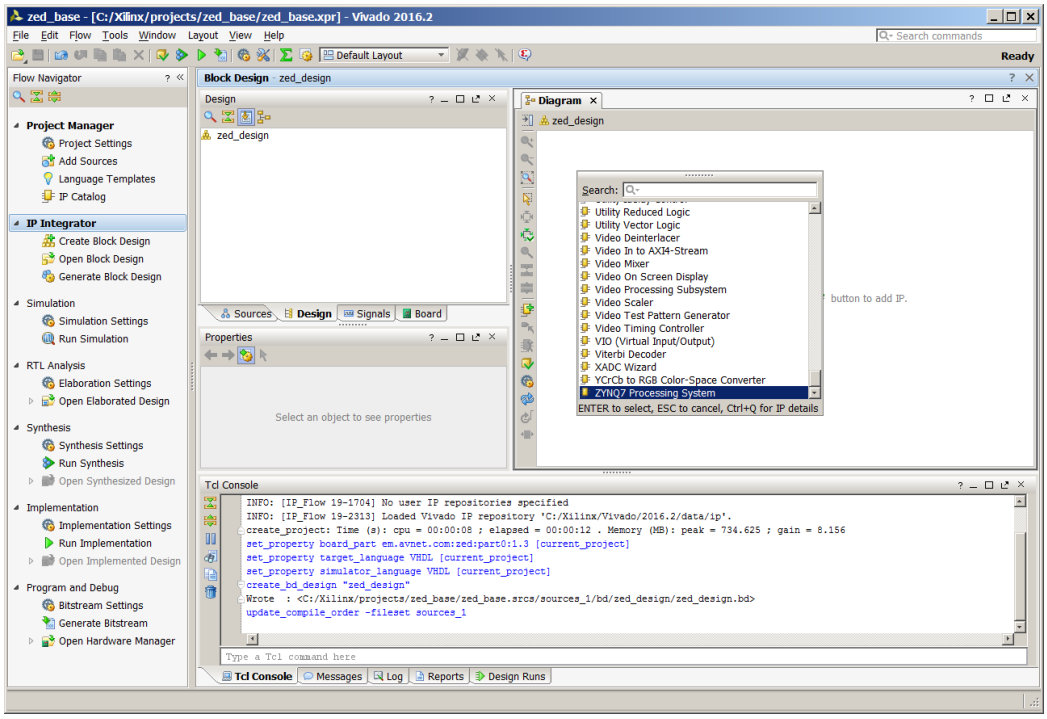
Name your block design

A small pop up will show up asking you to name your design. We will only have one block design in this project, so I have named mine 'zed_design'. Name your project and click OK.



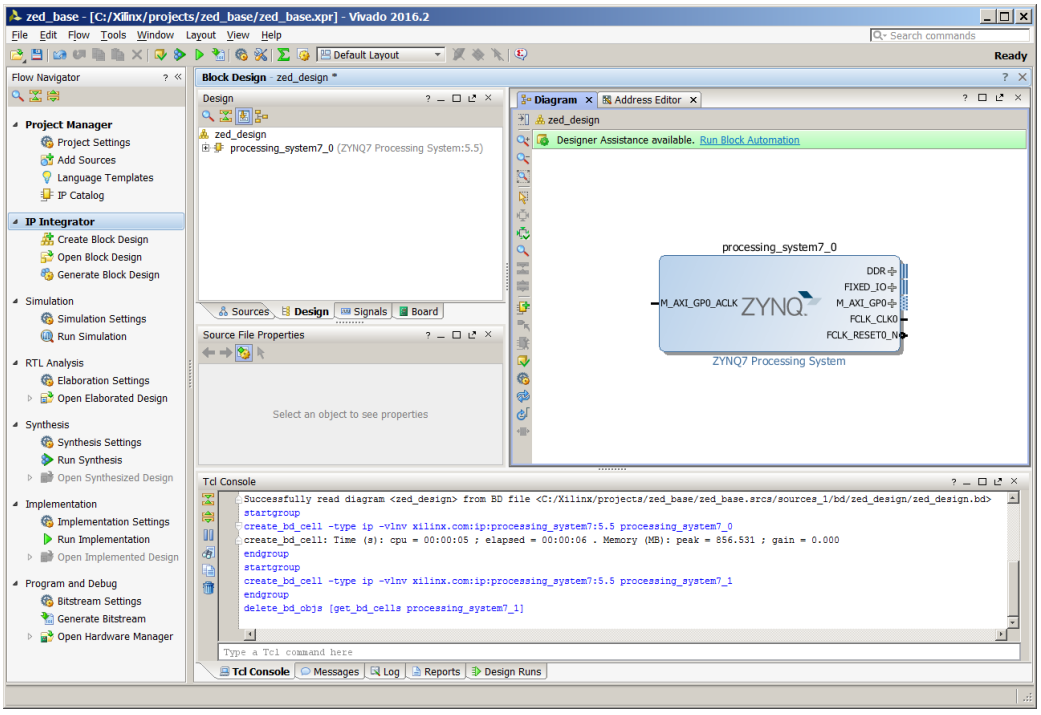
Opened blank IPI Block Design.

The IPI block design is now open, and you can see a "Diagram" tab now appears within the Block Design view on the right side of the Vivado window. Locate the small yellow button at the center Diagram tab. We want to add the Zynq Processing System (PS) to our design, so we will click the 'Add IP' button.



Add IP dialog list.

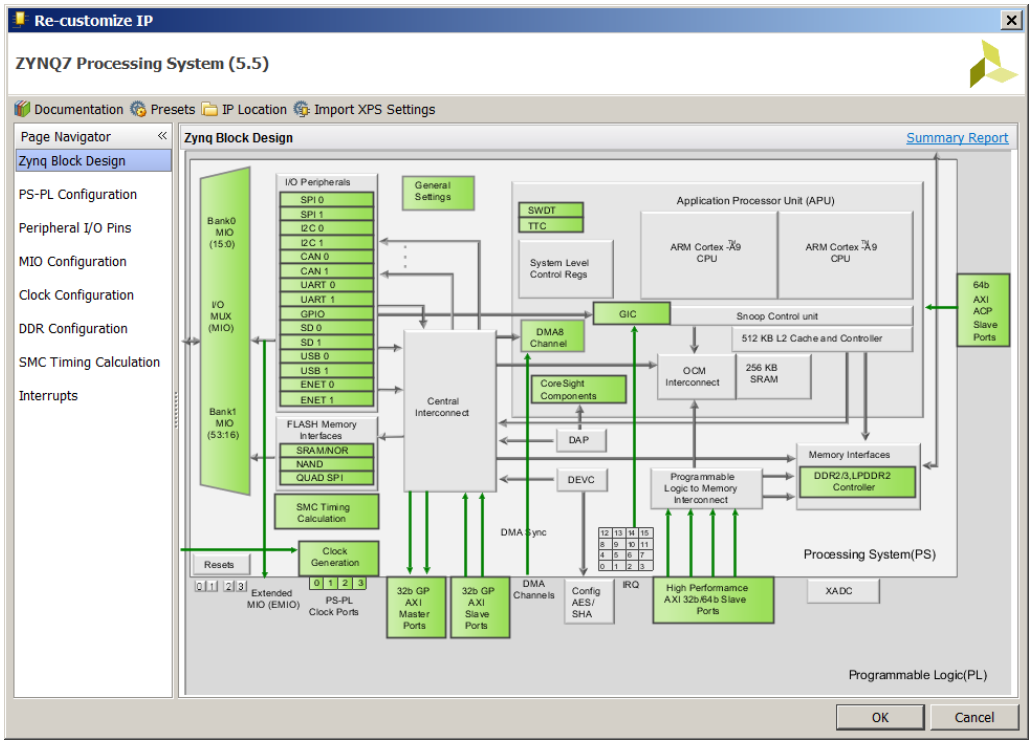
When we click the Add IP link, we are presented with a list of available IP to be added to the design. There are a large number, but we are only interested in one: "ZYNQ7 Processing System". Find the Zynq PS within the list (it's in alphabetical order) and double click to add it to your design.



Zynq PS within IPI.

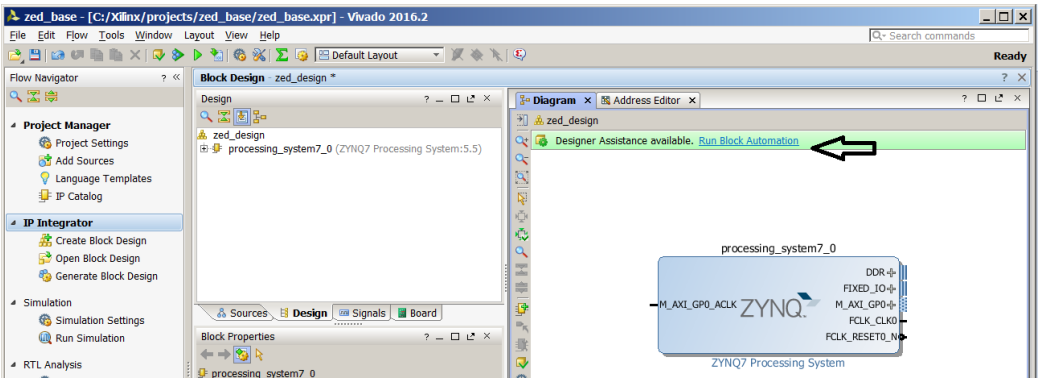
Awesome. We now have the PS within the IPI block. You can double click to view all of the settings you are used to seeing within XPS. Since we selected the Zedboard as our target, and

Vivado is 'Board Aware', we do not need to change any of the configuration. Here is what the window looks like though for your reference. You can poke around and just click Cancel when you are done (as to not save any changes).



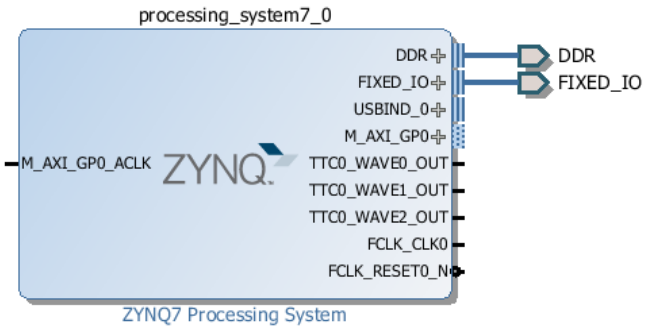
PS configuration tool - accessed via double clicking the PS block within IPI.

As we had previously stated, Vivado IPI is 'Board Aware'. This is really, REALLY, nice since this means you don't have to pull out each and every little single to the outside world (think about explicitly calling out all those DDR3 signals ... !). We can have the Vivado tool connect the signals that it knows are external by clicking the 'Run Block Automation' in the green advisory bar on the top of the Diagram tab window.



Vivado Board Aware'ness telling you it's okay to be lazy ...

Once you click the 'Run Block Automation' link, you will see that two busses have been defined on the top right of the PS block: DDR and FIZED_IO. The DDR bus is, rather explicitly, the DDR bus. The FIXED_IO bus is the MIO configuration for the Zedboard (since that was the board we targeted when creating the project). A pop-up will show asking if you want to 'auto connect' the two busses. Select OK to continue.

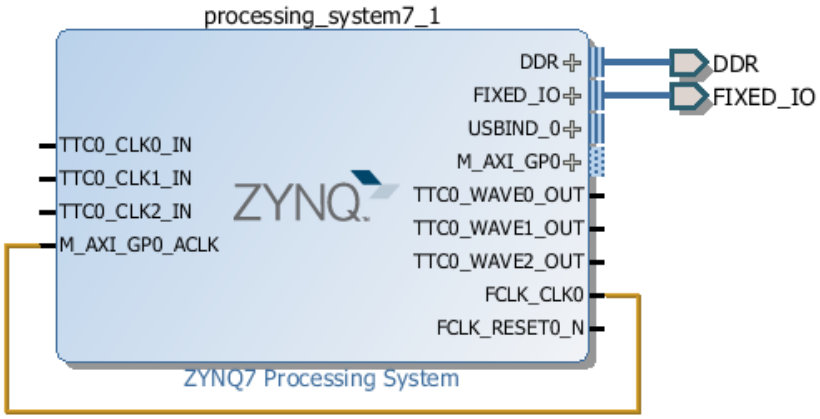


The result of the 'Run Block Automation' execution.

Now that we have the external IO connected, there is just a single other connection that we need to make. (Note: there is a few different ways we can do this, but for simplicity I am doing it this way, more on AXI busses and connections in a later blog post). We have a single AXI bus coming out of the PS (there are others, but we only have one configured here - a result of the 'board aware' configuration of the Zedboard), and we need to clock it with one of the four clocks that are produced via the PS. On the right side of the PS IPI block you can see the 'FCLK_CLK0' signal - this is one of the four output clocks from the PS. On the left side of the block you can see a 'M_AXI_GP0_ACLK' signal - this is the input clock for the single configured AXI bus on the PS. There are a number of other signals/busses present, however we won't be using any of those for our base Zynq design.

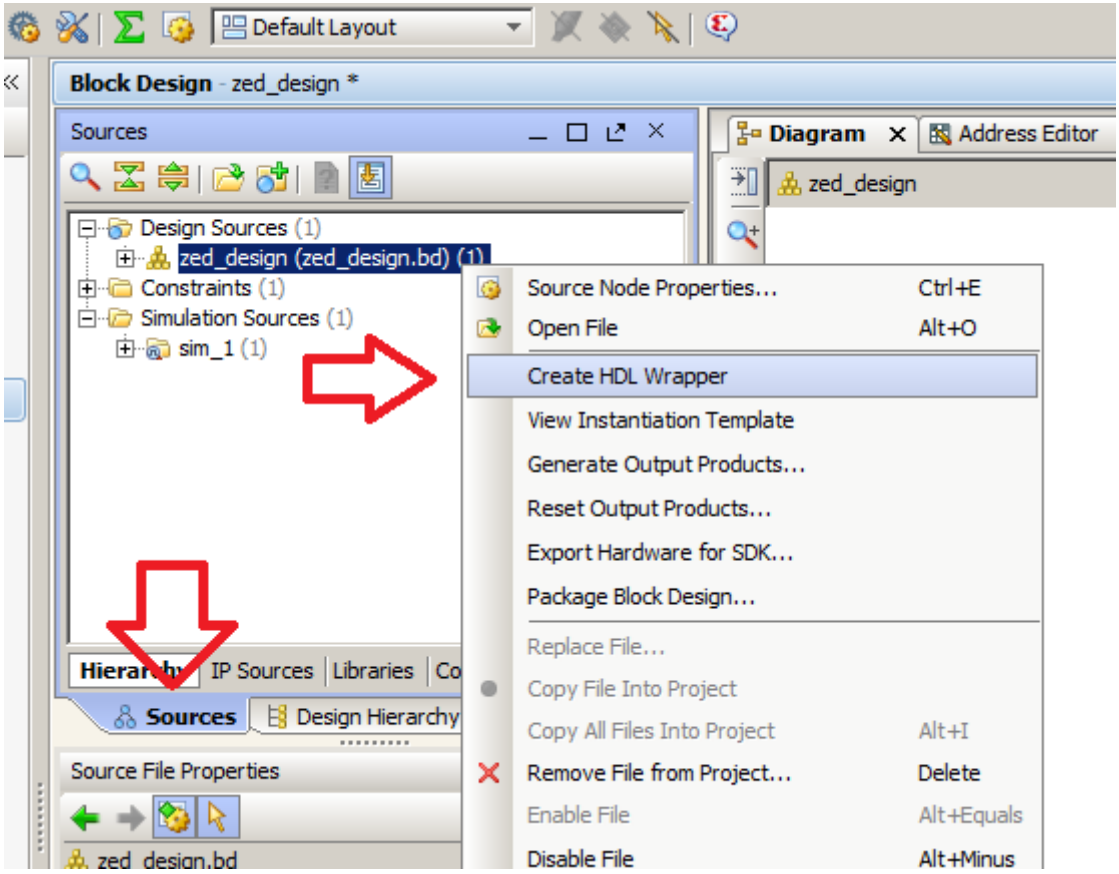
To connect signals we need to select a output signal (right side of a block) and then select a input signal (left side of a block). Move your mouse and select the FCLK_CLK0 signal, and then click the M_AXI_GP0_ACLK signal. You will notice after you select the output, the inputs on the left side of the block will highlight with green check markers (these will only show up if you hover over a block).

Note: if the 'line tool' stays engaged after you make the connection, you can hit 'ESC' on your keyboard to get out of the draw tool.



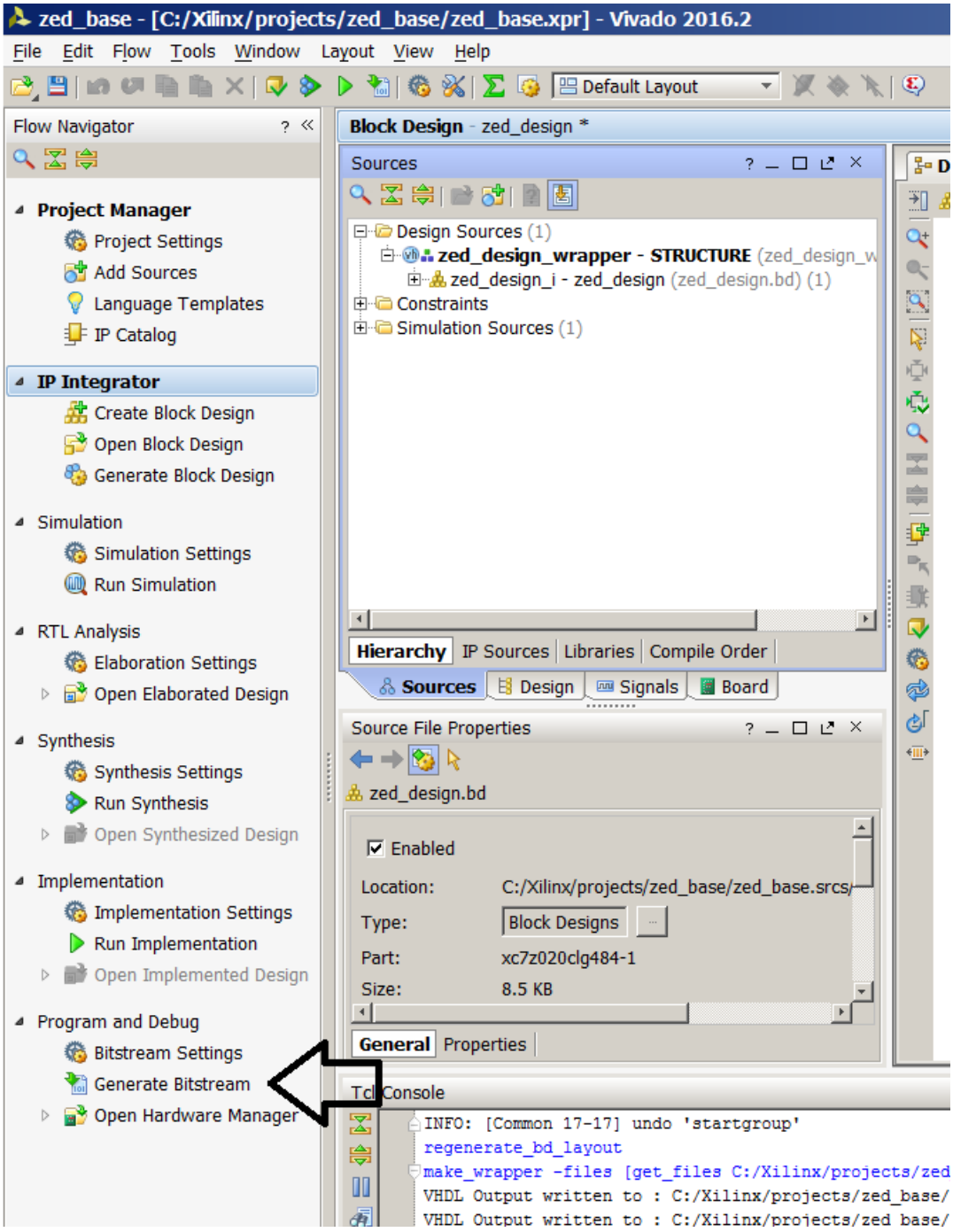
The connected, configured, and ports-made-external'ized PS system.

Boom, done. We've created our PS, configured it (okay Vivado did that for us but still ...), made the necessary ports external, and clocked our signal AXI port. Next we need to create an HDL wrapper (if you remember that from the PlanAhead XPS flow) for the Vivado synthesizer knows what to do with our IPI block.



View of creating the HDL wrapper.

Make sure you have the Sources tab selected, and then right mouse click on the zed_design IPI block and select Create HDL Wrapper. This will generate a HDL wrapper that the Vivado synthesizer understands. Once this happens, we are ready to generate our bitfile. This might sound like a large jump, but there isn't anything else in our design - it's almost entirely PS (the only PL portion is that AXI port support logic).



Within the Flow Navigator, find the 'Generated Bitstream' option.

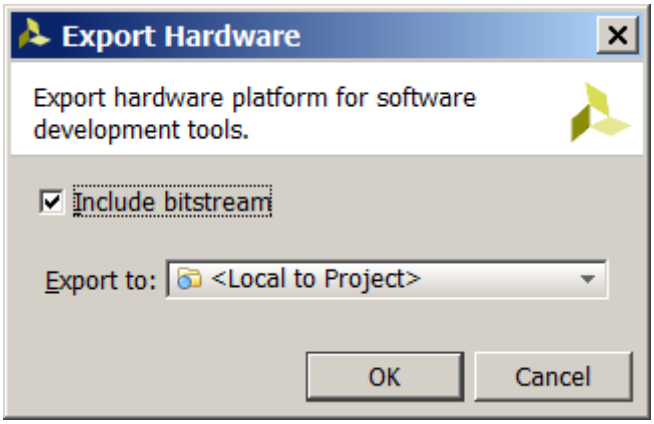
Depending on how much RAM you have, and your processor speeds (and number of cores) the processor generating the bitstream can take a few minutes to half an hour. On my machine it took about 7 minutes. Note that you will be presented with a pop up saying that you have

not run implementation ("There are no implementation results available . Okay to launch synthesis and implementation? ...". Select Yes to continue).

Vivado will do it's magic (I'm pretty sure --magic=enabled a valid switching into the compiler they use ...), and a pop-up will show asking you if you want to open the implemented design, View reports, Open hardware session, or Launch iMPACT. For this flow/example we will keep the default and 'Open Implemented Design'. Select the radio button and select OK.

After the implemented design is open (you will know this because a super awesome view of the Zynq AP SoC will be displayed in the right side of Vivado), we need to export the hardware design to SDK. We can do this using the File -> Export -> Export hardware for SDK. Note: if you don't see this option, you probably didn't open the implemented design. Open the implemented design by finding 'Open Implemented Design' under 'Implementation' within the Flow Navigator).

'Open Implemented Design' under 'Implementation' within the Flow Navigator).

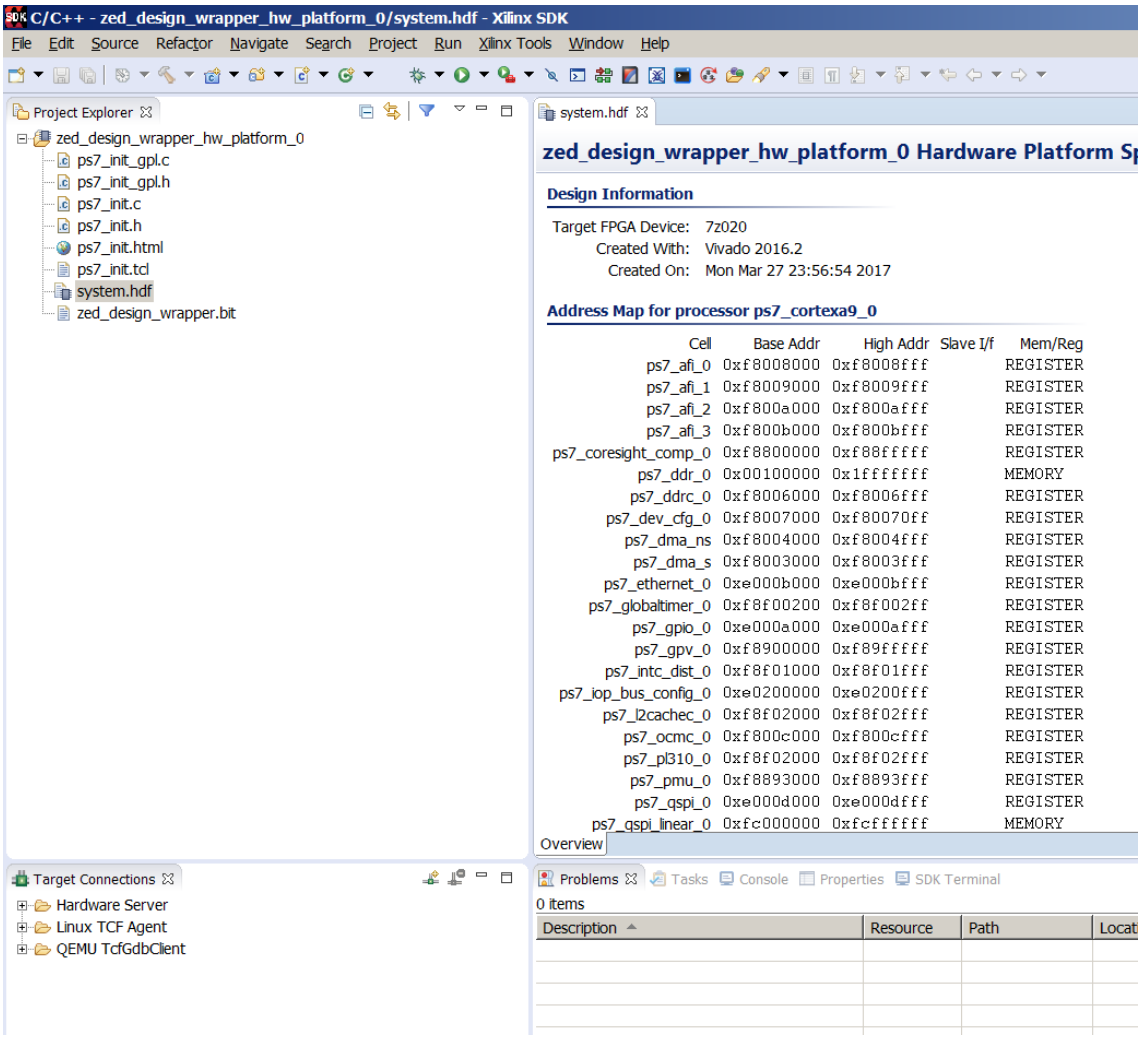


Export to SDK dialog.

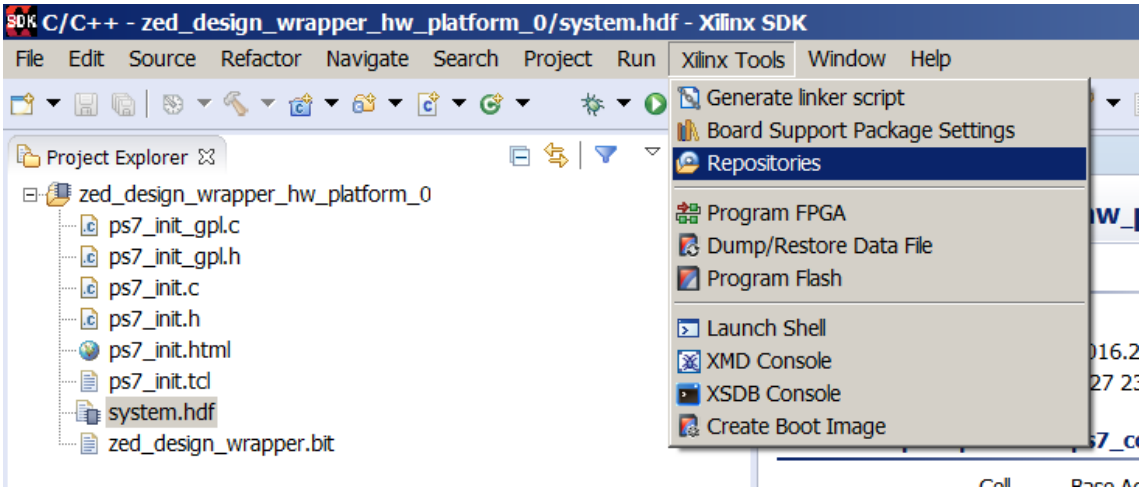
Make sure you select 'Export Hardware', 'Include bitstream', and 'Launch SDK', and leave all the other fields default and select OK.

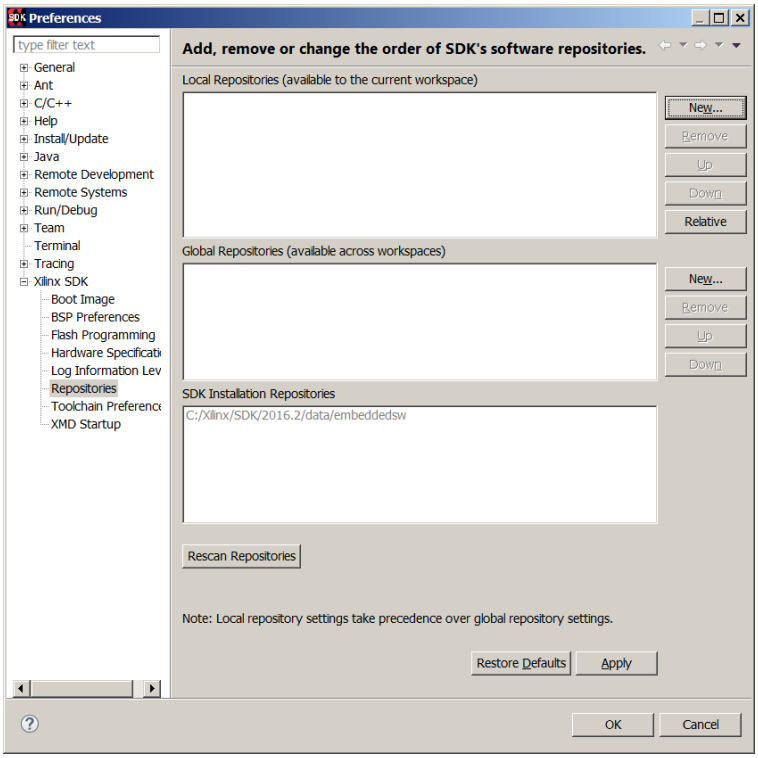
SDK: Software Development Kit

In Vivado: File → Launch SDK

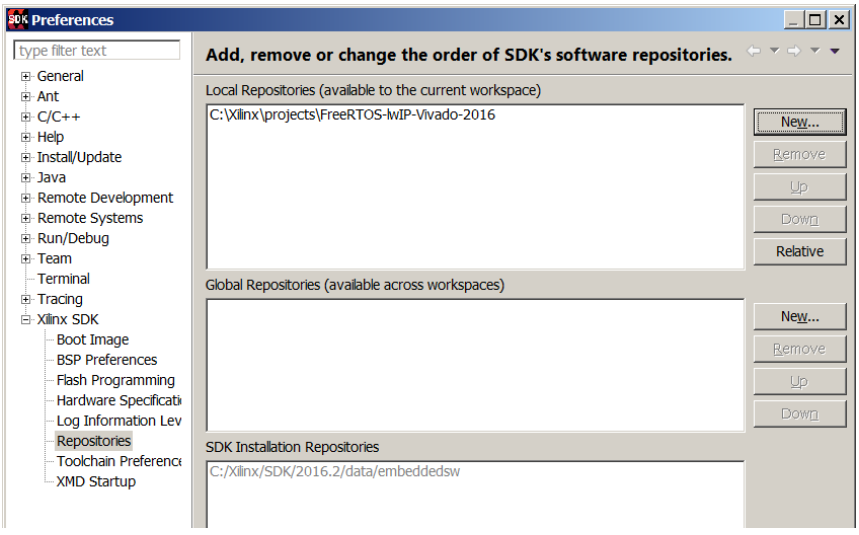


In SDK: Xilinx tools → Repostories



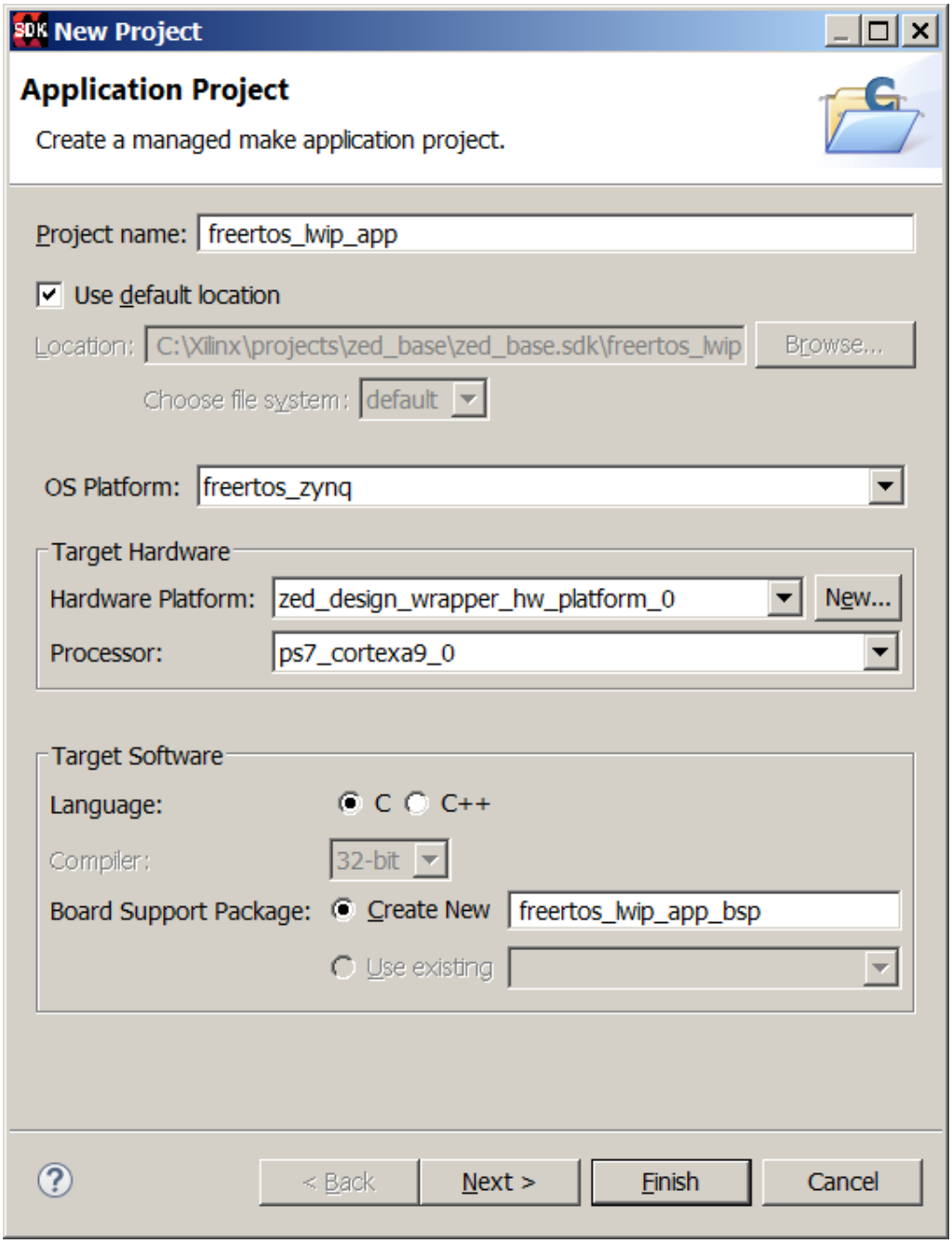


- Click **New** under *Local Repositories* section and give the path to **FreeRTOS-lwIP-Vivado-2016** directory.
- Click **Rescan** Repositories, then select **Apply** and then **OK**.
- This will ensure that the Xilinx SDK knows about the FreeRTOS and lwIP BSPs and the applications available to it.

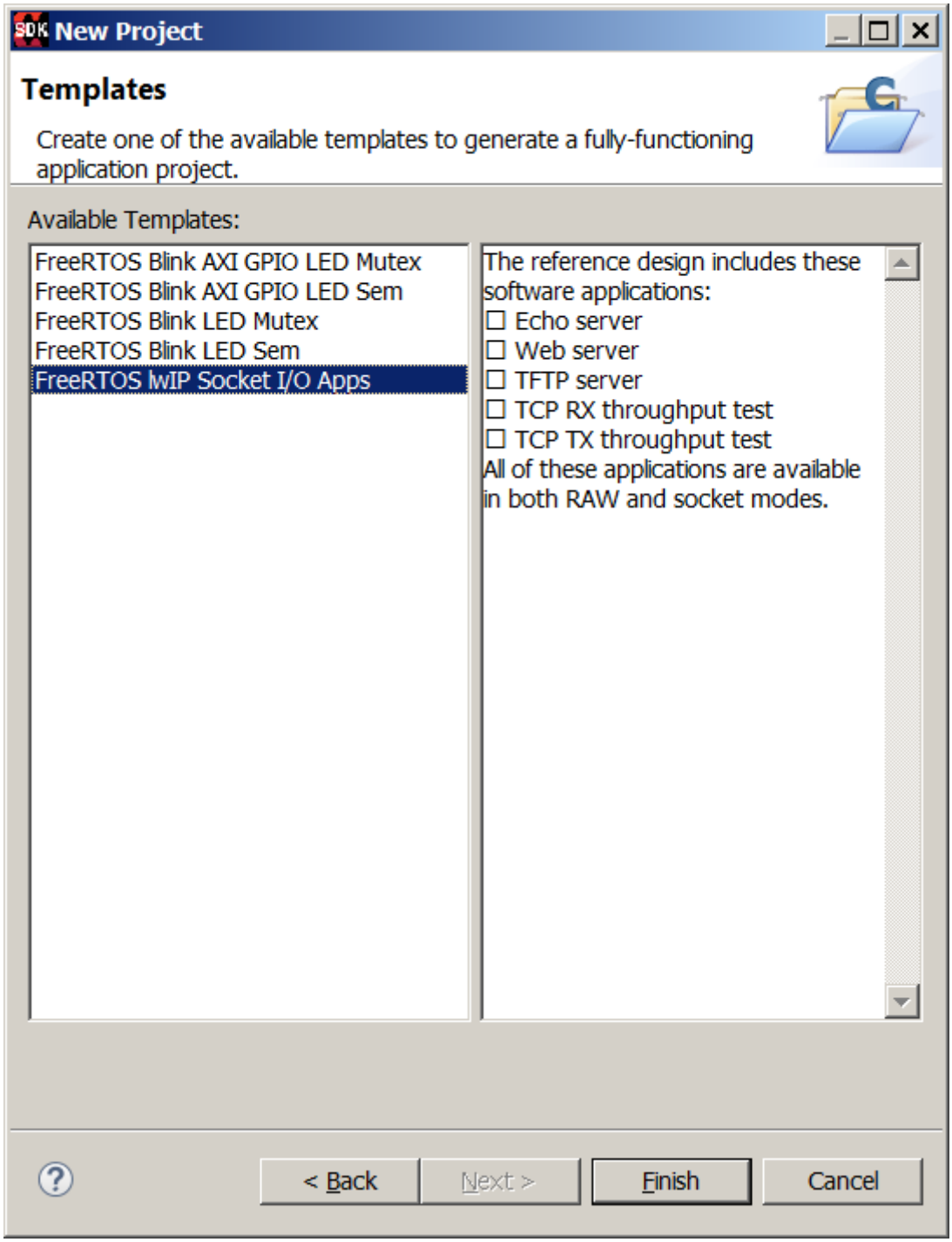


- Choose **File -> New -> Application Project** -> Select.
- New Application Project window opens up. Provide Project Name.
- Under Target Hardware tab, choose a Hardware Platform from dropdown list against Hardware Platform attribute. (Choice can be made to use pre-defined hardware platforms or create new hardware project, say zed_hw_platform)
- Choose the processor for which the application should be targeted.

- Under Target Software tab, select **freertos_zynq** as OS Platform. Name for Board Support Package will be populated based on the application project name. Accept the default or edit



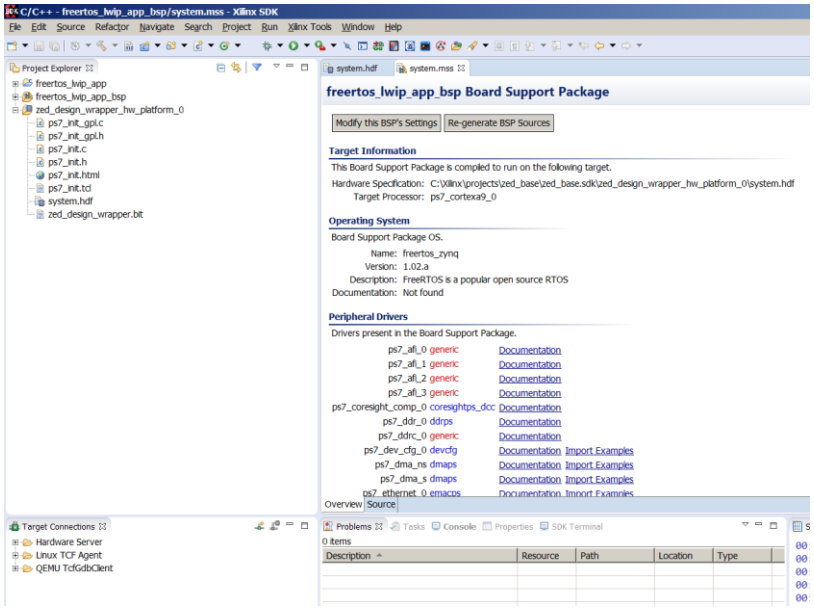
and **Click Next**.



Choose the FreeRTOS lwIP Soccket I/O apps Template and **Click Finish.**

The SDK Project

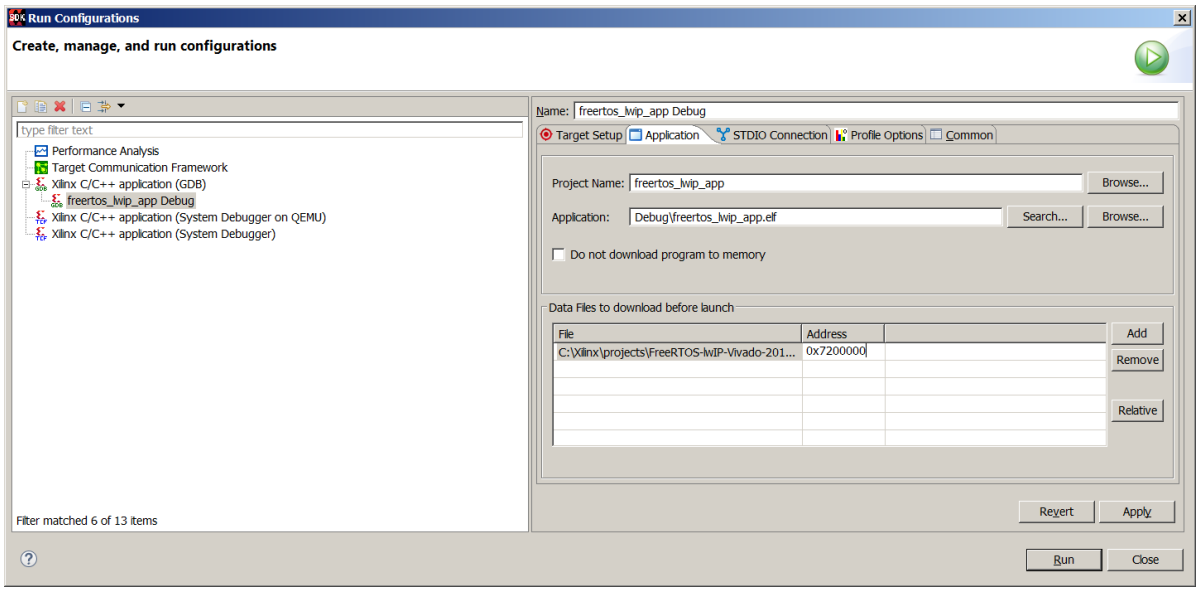
Three sdk projects – FreeRTOS lwIP socket application with the provided project name, board support package and Zedboard hardware platform project will be created.



Preparing the "Run"

Execute the lwIP FreeRTOS APPS

- *Choose Run -> Run Configurations …*
- Under **Xilinx C/ C++** application select **New** from the context menu and choose the lwip Socket app created earlier.
- In the **Application** tab click the "Add" button and browse to the location of the image.mfs file from the XAPP1026 files retrieved from the Xilinx website. Set the address to 0x7200000 and hit "Apply" then "Run".



- Once this is set up, future launches do not need this dialog. Just run from the run menu directly.
- From a browser on the same subnet navigate to fixed address as reported by the serial terminal output. The terminal will display information about the application.