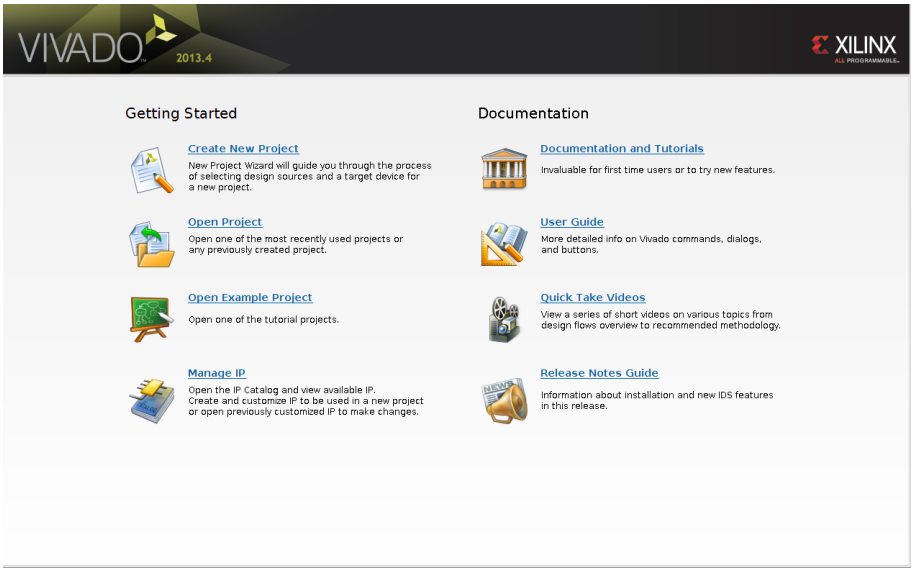

Figure 1: Fig 1

Hardware design for Zybo

In this lab we will use Xilinx Vivado to design hardware, it is easy to use and has all the programs we need built in.

Name your project, and select a place to put it, once that is done select RTL project and click next.

Click next for the next two dialogues, when asked to add a constraints file stop and lets add the one that we downloaded from Digilent.

This should tell Vivado about the hardware we are going to use. Next we need to tell Vivado what chip we are using, if we look back at the Digilent website for the Zybo we can make a note of the following information:

The ZYBO offers the following on-board ports and peripherals:

ZYNQ XC7Z2010-1CLG400C 512MB x32 DDR3 w/ 1050Mbps bandwidth Dual-role (Source/Sink) HDMI port 16-bits per pixel VGA output port Trimode (1Gbit/100Mbit/10Mbit) Ethernet PHY MicroSD slot (supports Linux file system)

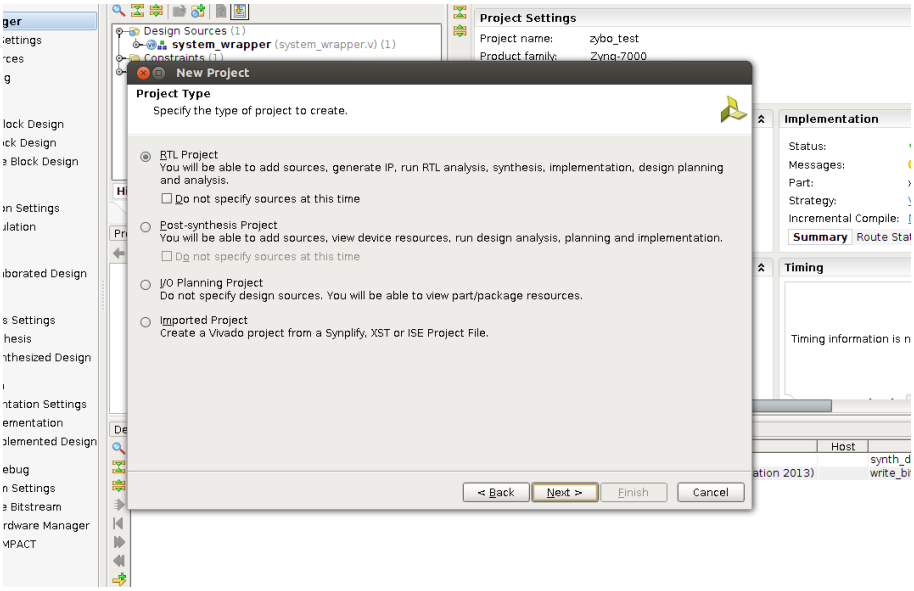

Figure 2: Fig 2

OTG USB 2.0 PHY (supports host and device)

The top line is what we want and will help us identify the chip. From the drop down menus select Zynq-7000 for Family, Zynq-7000 for sub family, clg400 for package, -1 for speed grade and C for temp grade. You will have two choices left xc7z010clg-400-1 and xc7z020clg400-1, choose the fist one since your Zynq chip is the Xilinx Zynq-7000 (Z-7010) as mentioned on the Digilent website. You also want to grab the hardware guide for Zybo, it will help in future posts if you are following along.

We are ready to confirm and create the project

So we should now have Vivado open with a new project like the picture below.

Now we are ready to create the block diagram and add some IP.

In the left side of the screen click on create block design. I named my block design system but I dont think the name really matters.

Now that we have a new block design, we can go ahead and add some IP to it. Click Add IP on the green highlight that appeared in the diagram window. Scroll down and select Zynq7 Processing System.

Press Enter and you should now see a Zynq processor on your block design.

So far so good, lets double-click the Zynq block and customize our IP to the Zybo.

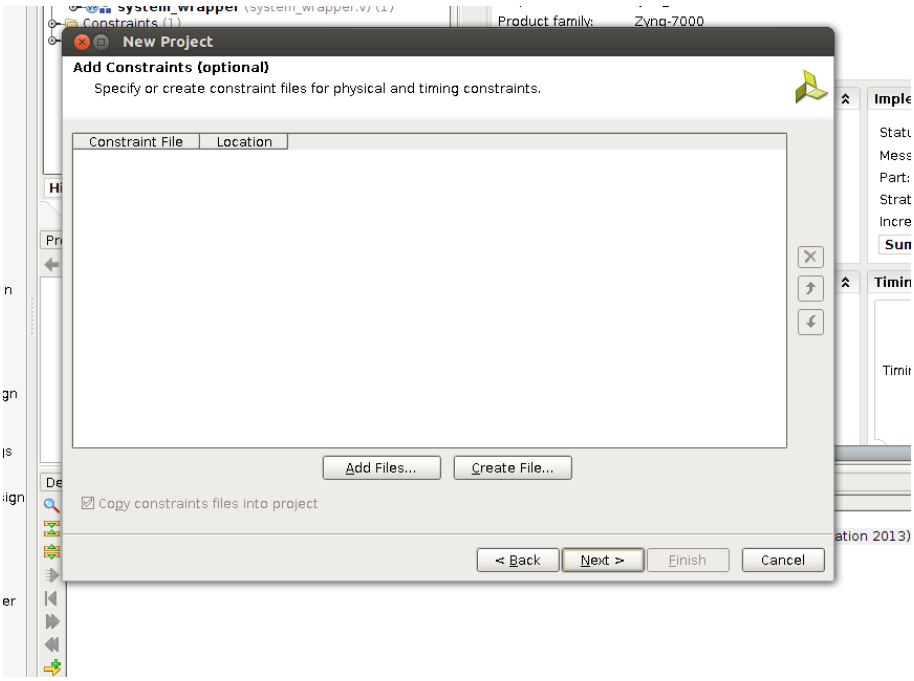

Figure 3: Fig 3

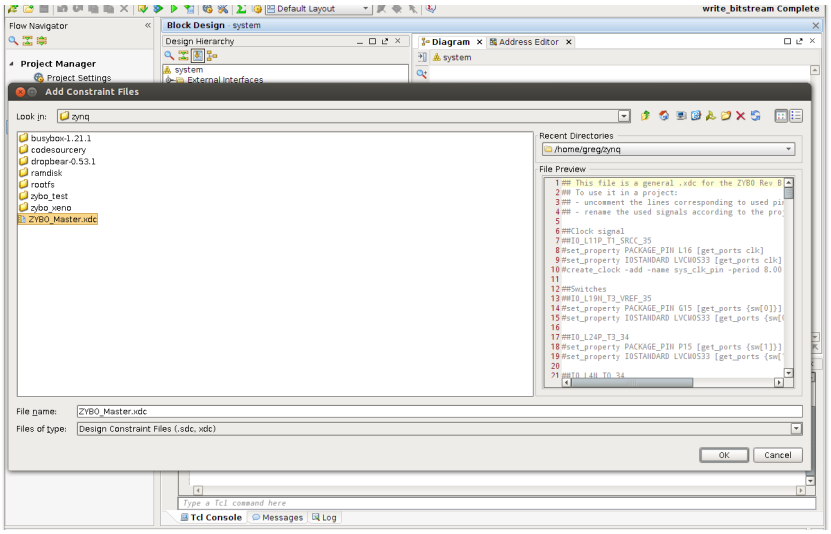

Figure 4: Fig 4

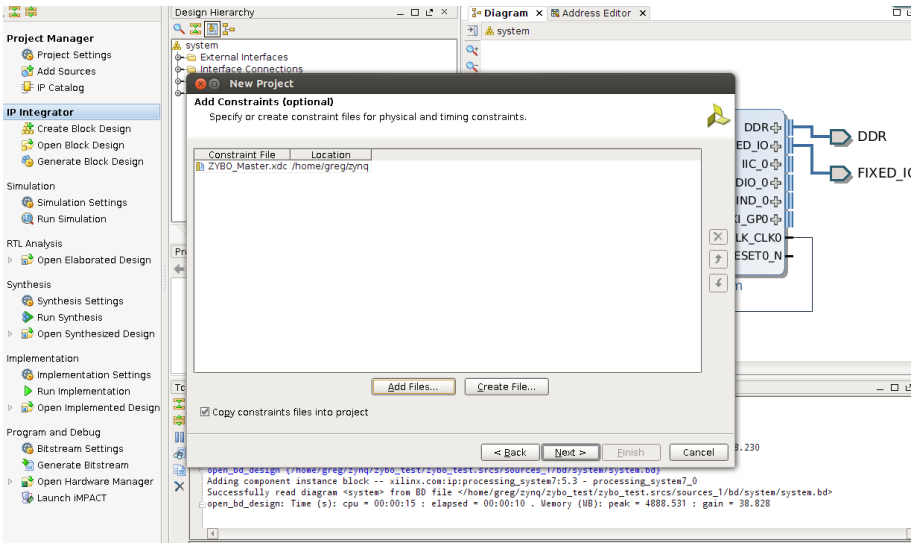

Figure 5: Fig 5

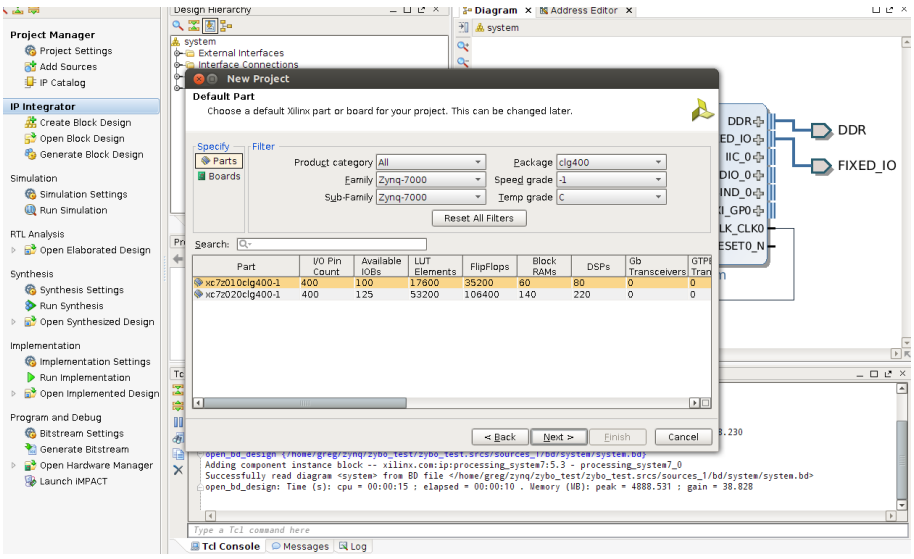

Figure 6: Fig 6

Now lets import the XPS settings that we downloaded from the Digilent site that will describe our hardware, click the import XPS settings button

Select the .xml file that we downloaded from Digilent, click OK. Now click OK in the import XPS settings window.

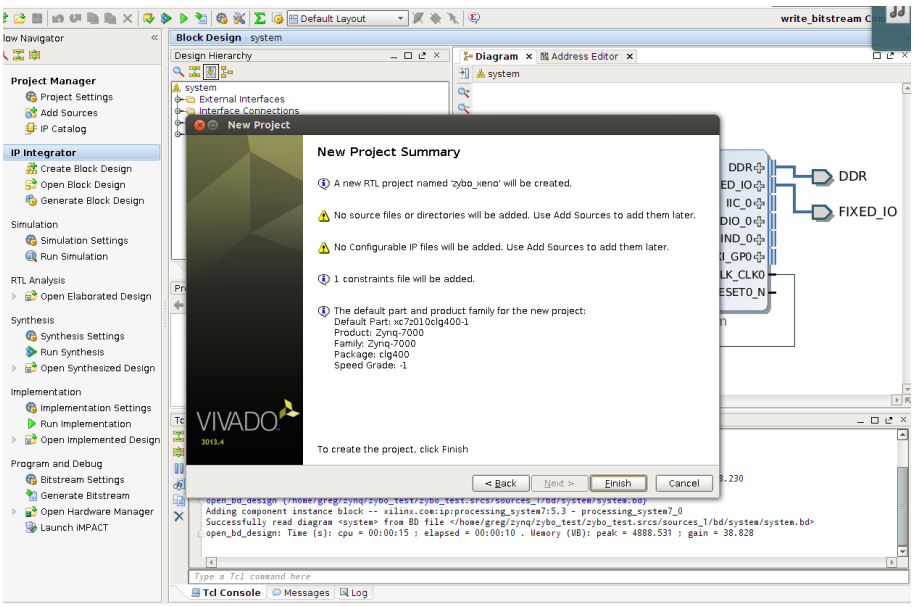

Figure 7: Fig 7

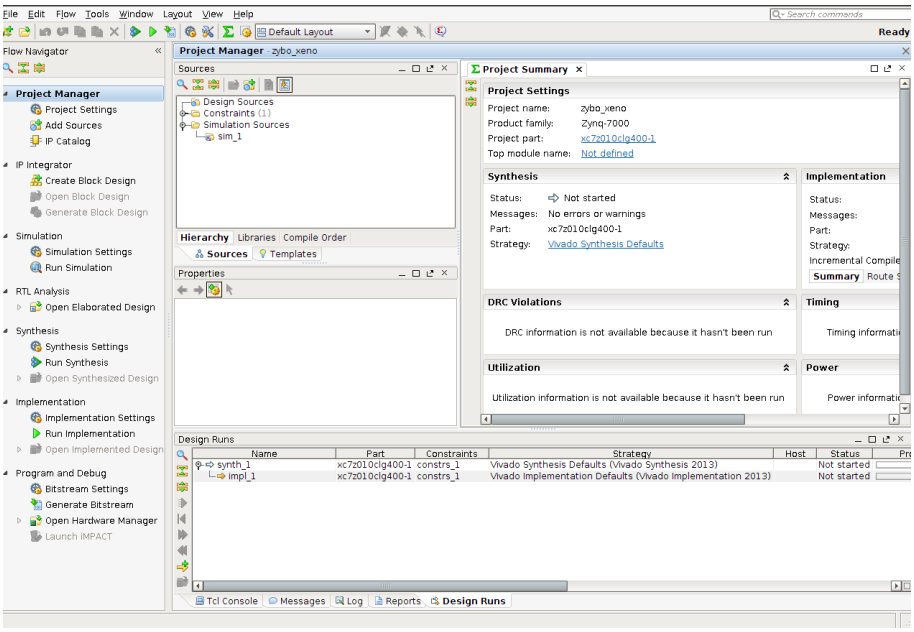

Figure 8: Fig 8

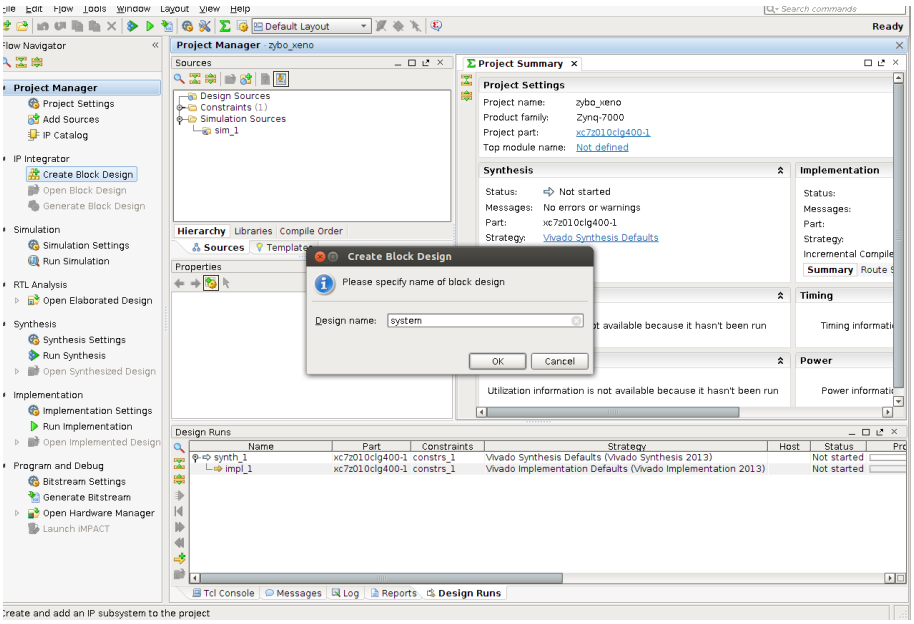

Figure 9: Fig 9

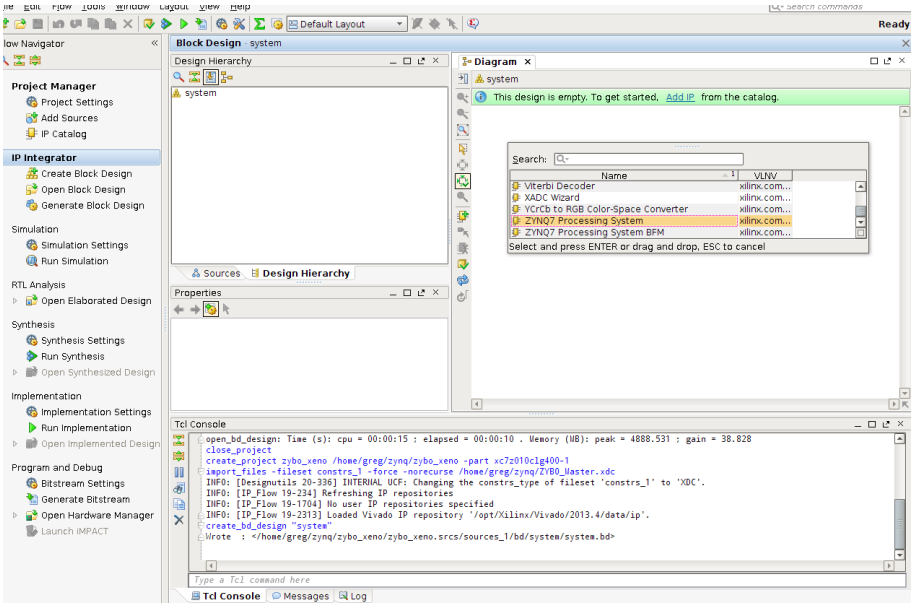

Figure 10: Fig 10

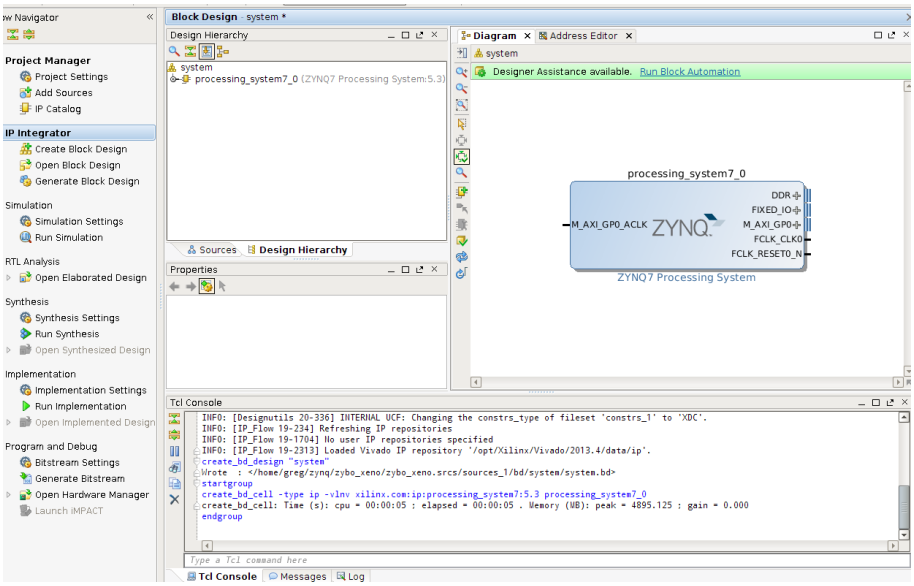

Figure 11: Fig 11

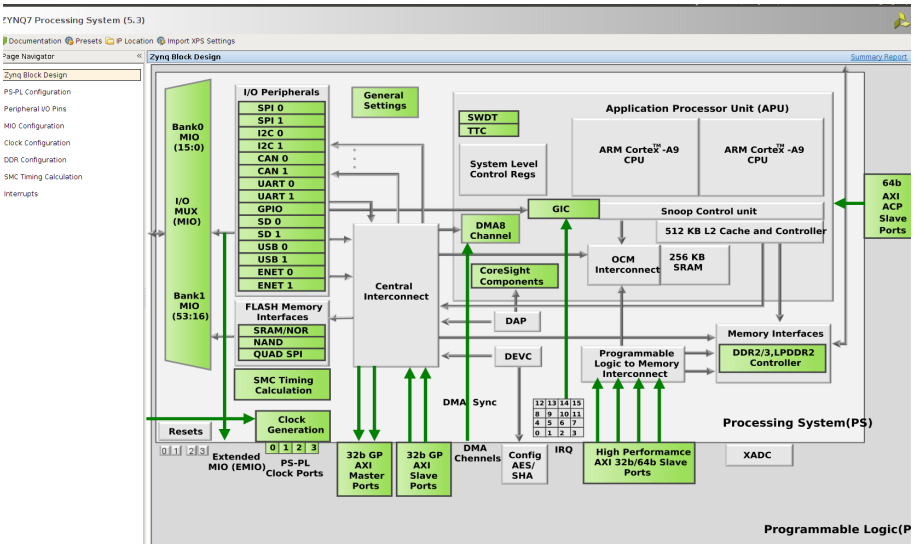

Figure 12: Fig 12

So we now see some check marks beside some peripherals. Lets take a second and look at the clock configuration. Click the clock configuration in the left side of the window, you should see something like the picture below.

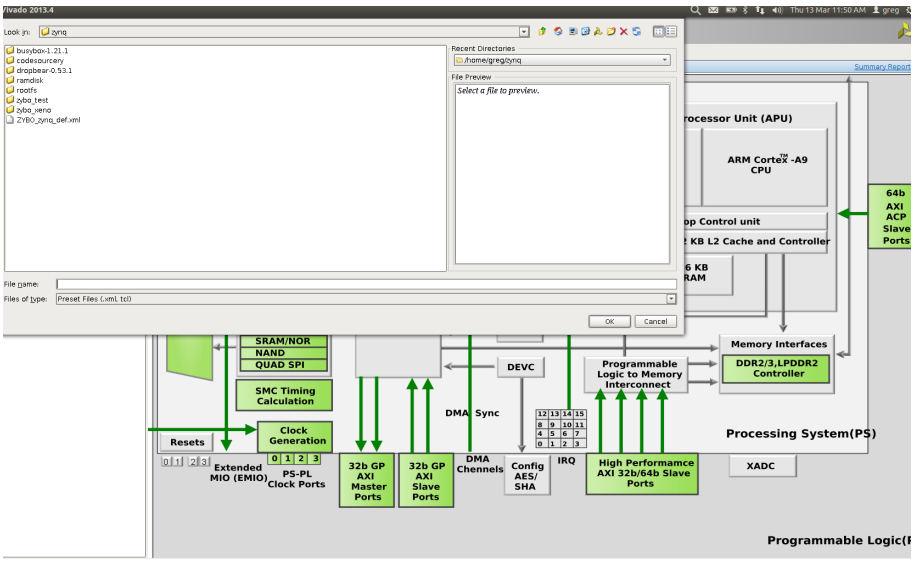

Figure 13: Fig 13

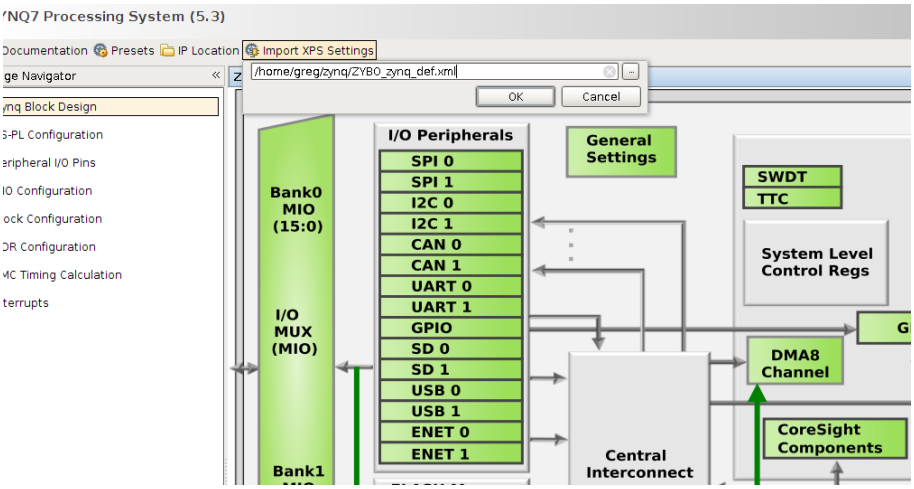

Figure 14: Fig 14

Make a note of the input frequency. This is DIFFERENT from both Zedboard and MicroZed and can cause some really frustrating problems when trying to add correct features to the device tree when we boot Linux. Ill explain what I ran into when we go over how to get Xenomai/Linux to boot. Click Ok and the customization screen should close and our processor should now have some inputs

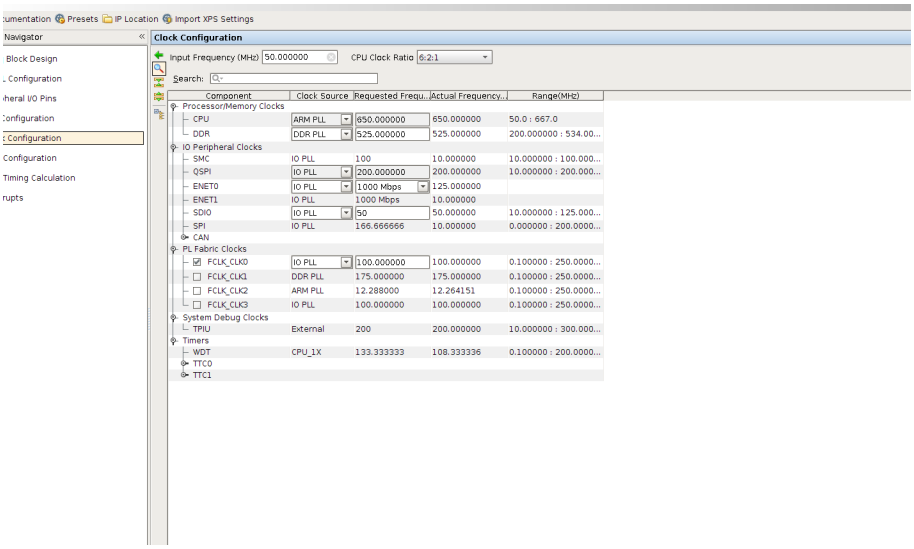

Figure 15: Fig 15

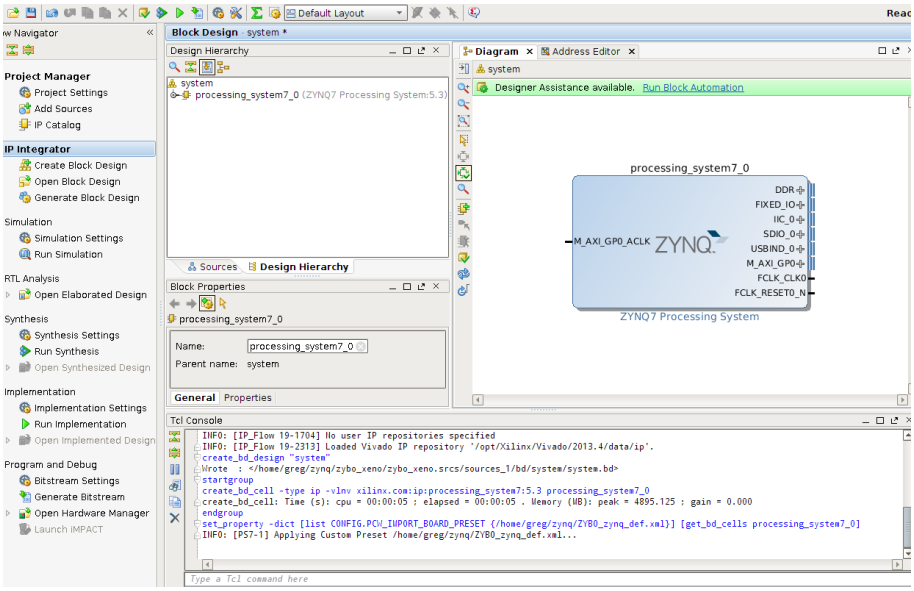

Figure 16: Fig 16

and outputs.

Connect the FCLK_CLK0 to the M_AXI_GP0_ACLK, once we scroll over the input a pencil appears and then connect each input similar to Labview if anyone

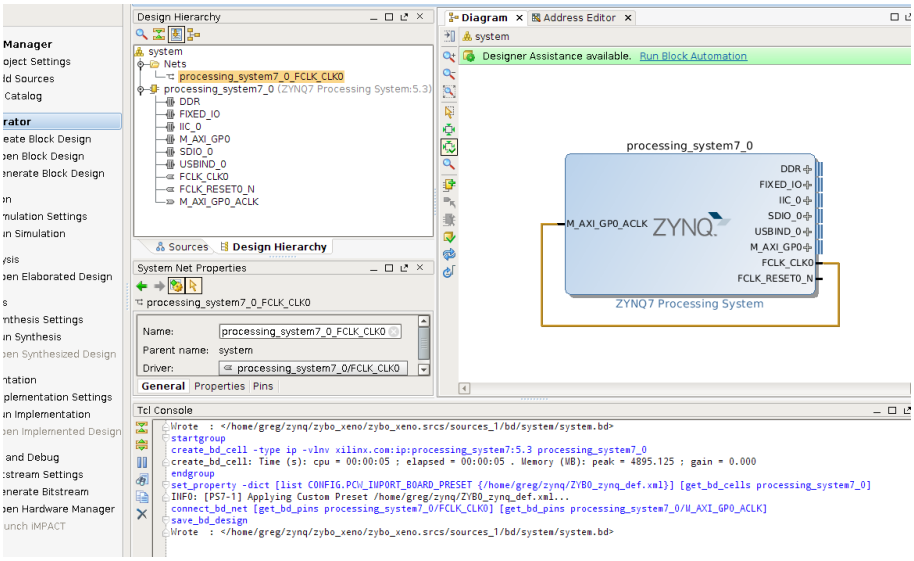

Figure 17: Fig 17

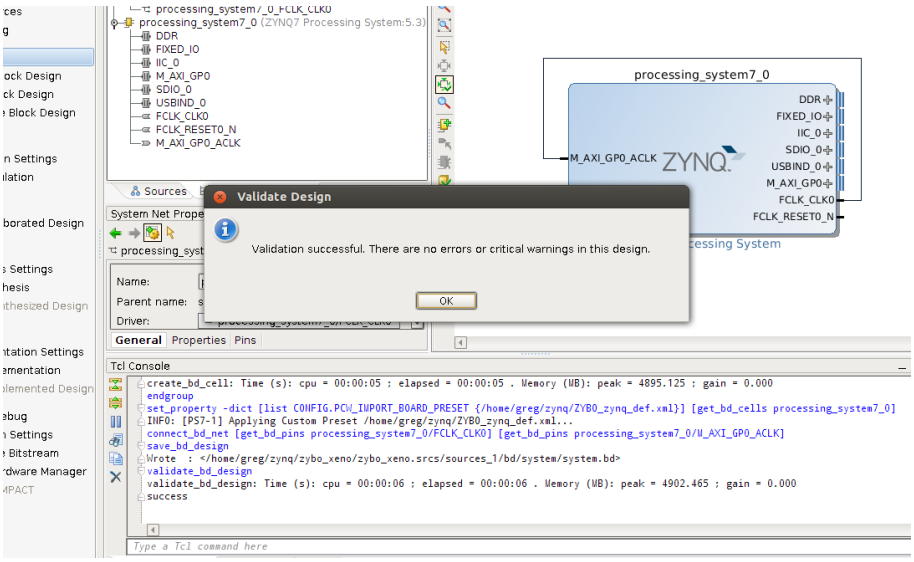

Figure 18: Fig 18

has used that before.

This pretty much just feeds a clock to the FPGA and is the most basic FPGA design we can do. Im not a FPGA expert and plan to use the Zybo to further my

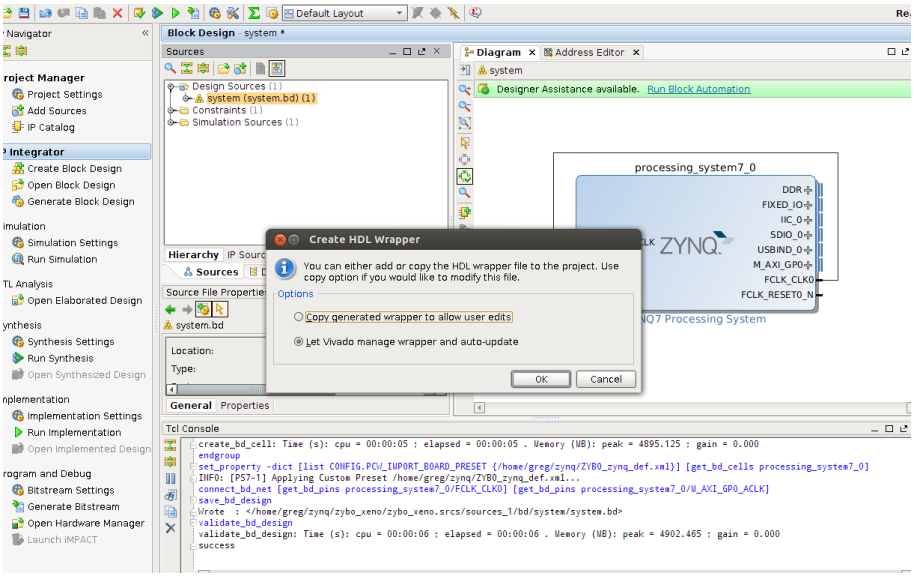

Figure 19: Fig 19

learning when it comes to FPGA design. I believe this pretty much brings the FPGA up and nothing else, so nothing on the FPGA is being used. Lets validate our design, before we start to create the HDL wrappers and bit file. Run the Block Automation as suggested by the green highlight.

Click the sources tab on the block design and right-click the system.bd file and select Create HDL Wrapper.

Once that is complete we should see some Verilog or VHDL files. Now we can go ahead and generate the bitstream file, this should be on the left side of the screen near the bottom.

Once we are done, we can open the implemented design.

We are pretty much done! The next step is to export our design to the Xilinx SDK to create the first stage bootloader. This will be the subject of my next post. Remember to save your project since well need it in my next post.

If anyone runs into problems let me know I may have a step or two out-of-order, but I was able to create the bit file again following these steps. Questions and comments are always welcome.

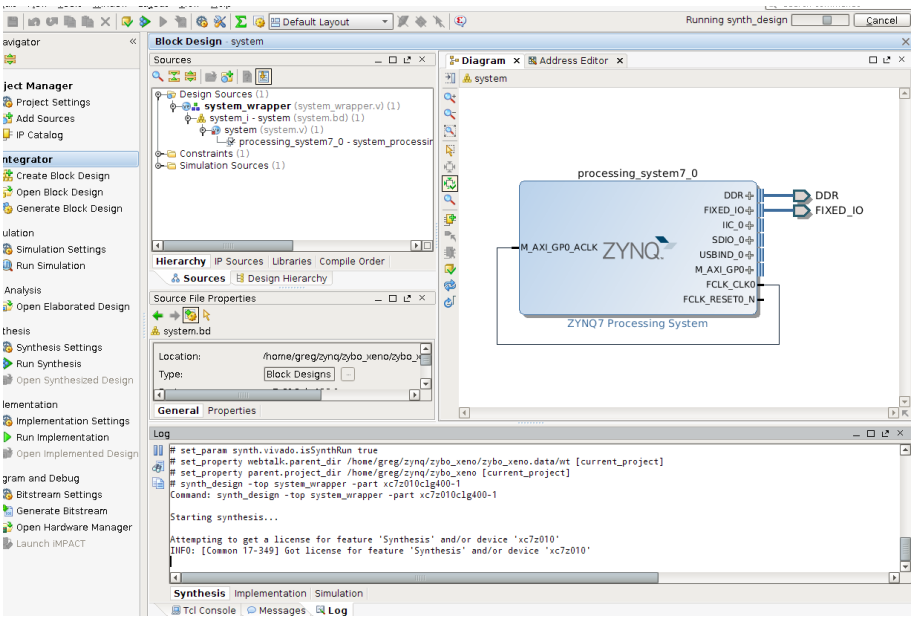

Figure 20: Fig 20

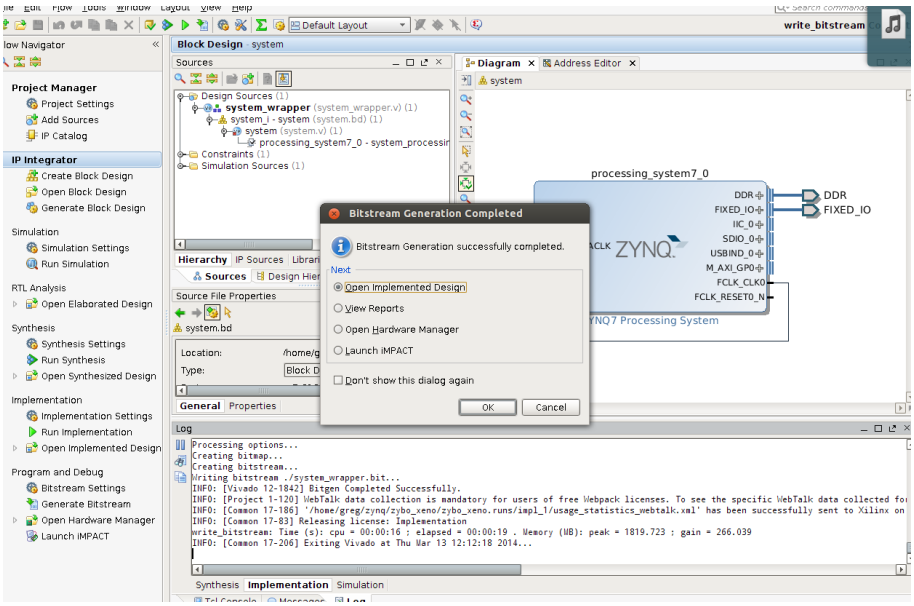

Figure 21: Fig 21