

Setting up the Ultra96 Baseline Xilinx Environment

This document describes how to set up the Xilinx environment. Here are 3 key web pages to review before starting:

LRC Accounts: (<https://wikis.utexas.edu/display/eceit/User+Accounts>)

LRC machines <https://wikis.utexas.edu/display/eceit/ECE+Linux+Application+Servers>

Log on: <https://wikis.utexas.edu/display/eceit/Connecting+to+Linux+Application+Servers>

Once you have successfully logged into an LRC machine, execute the following command:

```
module avail xilinx
```

You will see the following:

```
----- /home/projects/Modules/tools -----
xilinx/2018          xilinx/2016(default) xilinx/2019
xilinx/2010          xilinx/2017          xilinx/2020
xilinx/2012          xilinx/2017-rh7      xilinx/2022
xilinx/2014          xilinx/2017.4
xilinx/2014.2        xilinx/2017a
```

Execute the following commands:

```
module load xilinx/2022
which vivado
```

You should see the following response:

```
/usr/local/packages/xilinx_2022/Vivado/2022.2/bin/vivado
```

Execute the following commands:

```
cd /misc/scratch
mkdir <your login name>
cd <your login name>
```

Execute the following commands:

```
cp /home/projects/courses/spring_24/ee382n4-17365/arch/labs/BASELINE_SP_2024.tar.gz .
tar zxvf BASELINE_SP_2024.tar.gz
ls -l
```

You will see the following directory elements:

```
ip_repo
CONSTRAINTS
BASELINE_SP_2024/
```

Change directory into the BASELINE_SP_2024 directory:

```
cd BASELINE_SP_2024/
```

You will see the following directory elements:

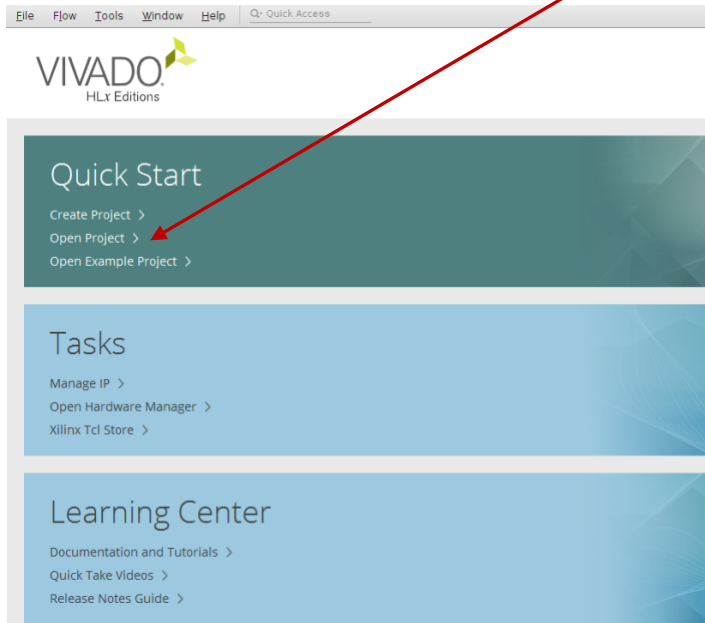
```
ultra96v2_oob          ultra96v2_oob.cache    ultra96v2_oob.hw
ultra96v2_oob.ip_user_files  ultra96v2_oob.runs    ultra96v2_oob.sim
ultra96v2_oob.srcs        ultra96v2_oob.xpr
```

Setting up the Ultra96 Baseline Xilinx Environment

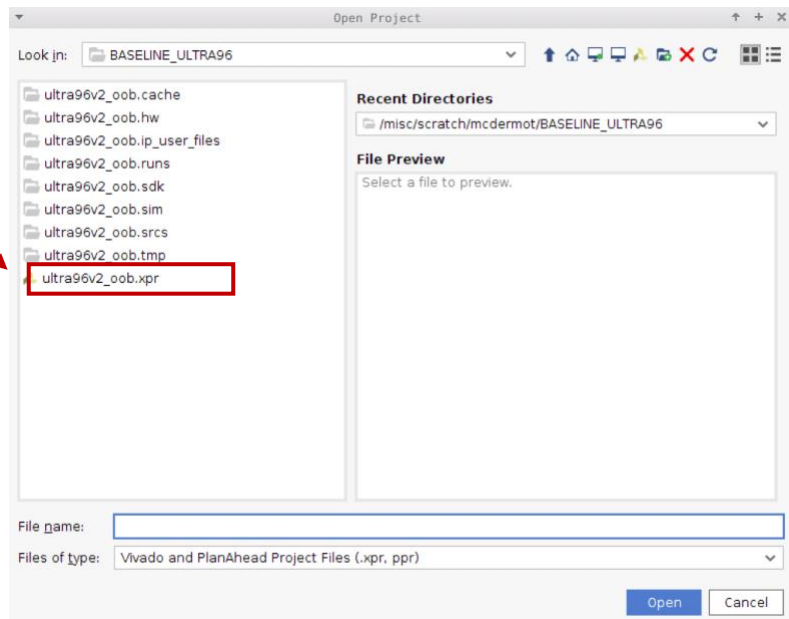
Execute the following command:

vivado &

The following window be displayed. Select: **Open Project**



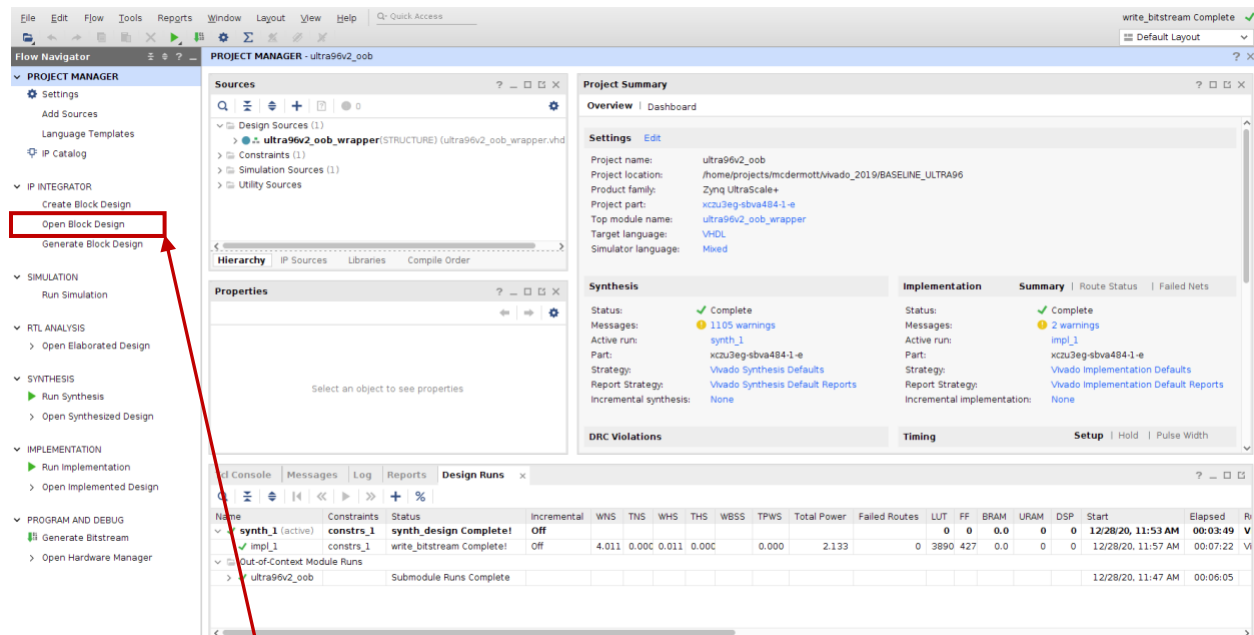
Select **ultra96v2_oob.xpr**



NOTE: you can also execute **vivado ultra96v2_oob.xpr &**

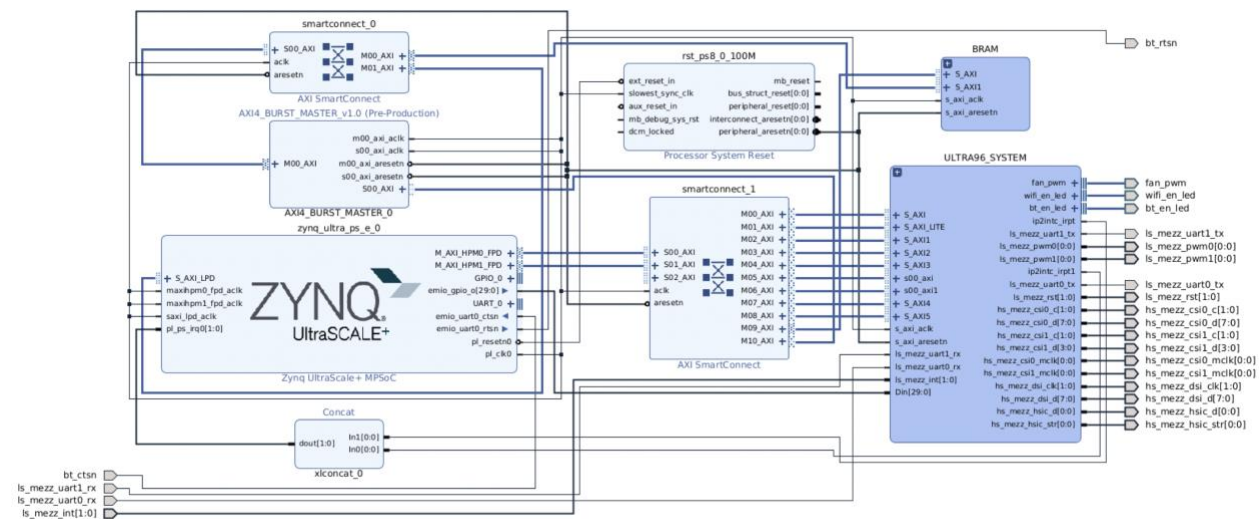
Setting up the Ultra96 Baseline Xilinx Environment

You will see the following display once Vivado is up and running. Vivado runs very slow on overloaded machines in the LRC.



Select **Open Block Design**

You will see something displayed that looks like the following schematic:

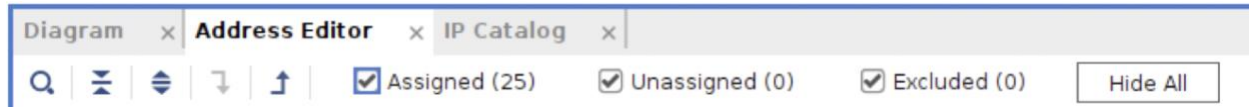


This is the baseline schematic that you will start with for the Lab assignments. Note that this schematic contains two hierarchical elements highlighted in dark blue. This is used to de-clutter the main schematic.

SIDE NOTE: Every time you change and reopen a Vivado schematic, it will look different.

Setting up the Ultra96 Baseline Xilinx Environment

Select **Address Editor** and you will see the addresses of the components attached to the Processing System (PS). **Note** that there are 40 address bits in this architecture.



You will see the following address mapping:

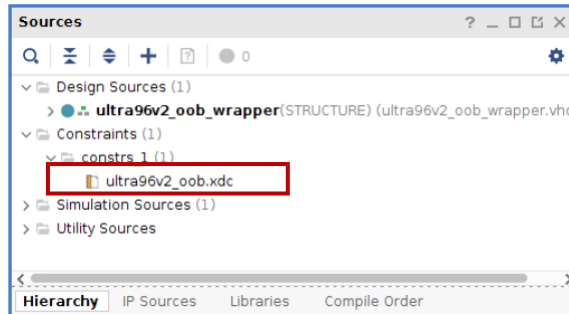
Name	Interface	Slave Segment	Master Base Address	Range	Master High Address
AXI4_BURST_MASTER_0					
/AXI4_BURST_MASTER_0/M00_AXI (40 address bits : 1T)					
/Block_Memory/axi_bram_ctrl_1/S_AXI	S_AXI	Mem0	0x00_C000_0000	8K	0x00_C000_1FFF
/zynq_ultra_ps_e_0/SAXIGP6	S_AXI_LPD	LPD_LPS_OCM	0x00_FF80_0000	8M	0x00_FFFF_FFFF
/zynq_ultra_ps_e_0/SAXIGP6	S_AXI_LPD	.PD_DDR_LOW	0x00_4000_0000	1G	0x00_7FFF_FFFF
zynq_ultra_ps_e_0					
/zynq_ultra_ps_e_0/Data					
/AXI4_BURST_MASTER_0/S00_AXI	S00_AXI	S00_AXI_reg	0x00_A000_0000	4K	0x00_A000_0FFF
/AXI4_BURST_MASTER_0/S00_AXI	S00_AXI	S00_AXI_reg	0x00_A000_0000	4K	0x00_A000_0FFF
/Block_Memory/axi_bram_ctrl_0/S_AXI	S_AXI	Mem0	0x00_A000_2000	8K	0x00_A000_3FFF
/Block_Memory/axi_bram_ctrl_0/S_AXI	S_AXI	Mem0	0x00_A000_2000	8K	0x00_A000_3FFF
/ULTRA96_SYSTEM/BD_CTL_GPIO/axi_gpio_0/S_AXI	S_AXI	Reg	0x00_A001_0000	64K	0x00_A001_FFFF
/ULTRA96_SYSTEM/BD_CTL_GPIO/axi_gpio_0/S_AXI	S_AXI	Reg	0x00_A001_0000	64K	0x00_A001_FFFF
/ULTRA96_SYSTEM/BD_CTL_GPIO/axi_gpio_1/S_AXI	S_AXI	Reg	0x00_A002_0000	64K	0x00_A002_FFFF
/ULTRA96_SYSTEM/BD_CTL_GPIO/axi_gpio_1/S_AXI	S_AXI	Reg	0x00_A002_0000	64K	0x00_A002_FFFF
/ULTRA96_SYSTEM/Low_Speed_MEZZ/PWM_w_Int_0/s00_axi	s00_axi	reg0	0x00_A003_0000	64K	0x00_A003_FFFF
/ULTRA96_SYSTEM/Low_Speed_MEZZ/PWM_w_Int_0/s00_axi	s00_axi	reg0	0x00_A003_0000	64K	0x00_A003_FFFF
/ULTRA96_SYSTEM/Low_Speed_MEZZ/PWM_w_Int_1/s00_axi	s00_axi	reg0	0x00_A004_0000	64K	0x00_A004_FFFF
/ULTRA96_SYSTEM/Low_Speed_MEZZ/PWM_w_Int_1/s00_axi	s00_axi	reg0	0x00_A004_0000	64K	0x00_A004_FFFF
/ULTRA96_SYSTEM/Low_Speed_MEZZ/axi_gpio_2/S_AXI	S_AXI	Reg	0x00_A005_0000	64K	0x00_A005_FFFF
/ULTRA96_SYSTEM/Low_Speed_MEZZ/axi_gpio_2/S_AXI	S_AXI	Reg	0x00_A005_0000	64K	0x00_A005_FFFF
/ULTRA96_SYSTEM/Low_Speed_MEZZ/axi_uart16550_0/S_AXI	S_AXI	Reg	0x00_A006_0000	64K	0x00_A006_FFFF
/ULTRA96_SYSTEM/Low_Speed_MEZZ/axi_uart16550_0/S_AXI	S_AXI	Reg	0x00_A006_0000	64K	0x00_A006_FFFF
/ULTRA96_SYSTEM/Low_Speed_MEZZ/axi_uart16550_1/S_AXI	S_AXI	Reg	0x00_A007_0000	64K	0x00_A007_FFFF
/ULTRA96_SYSTEM/Low_Speed_MEZZ/axi_uart16550_1/S_AXI	S_AXI	Reg	0x00_A007_0000	64K	0x00_A007_FFFF
/ULTRA96_SYSTEM/SYS_MGMT/axi_gpio_3/S_AXI	S_AXI	Reg	0x00_A008_0000	64K	0x00_A008_FFFF
/ULTRA96_SYSTEM/SYS_MGMT/axi_gpio_3/S_AXI	S_AXI	Reg	0x00_A008_0000	64K	0x00_A008_FFFF
/ULTRA96_SYSTEM/SYS_MGMT/system_management_wiz_0/S_AXI_LITE	S_AXI_LITE	Reg	0x00_A009_0000	64K	0x00_A009_FFFF
/ULTRA96_SYSTEM/SYS_MGMT/system_management_wiz_0/S_AXI_LITE	S_AXI_LITE	Reg	0x00_A009_0000	64K	0x00_A009_FFFF

There are two bus masters in this Baseline design. The ZYNQ_ULTRA is the Processing System (PS) and the AXI4_BURST_MASTER is in the Programmable Logic (PL). The AXI4_BURST_MASTER will be initially used to run memory tests in the PS and PL. Ultimately it will be used in the class project to transfer data to-and-from the PS memory.

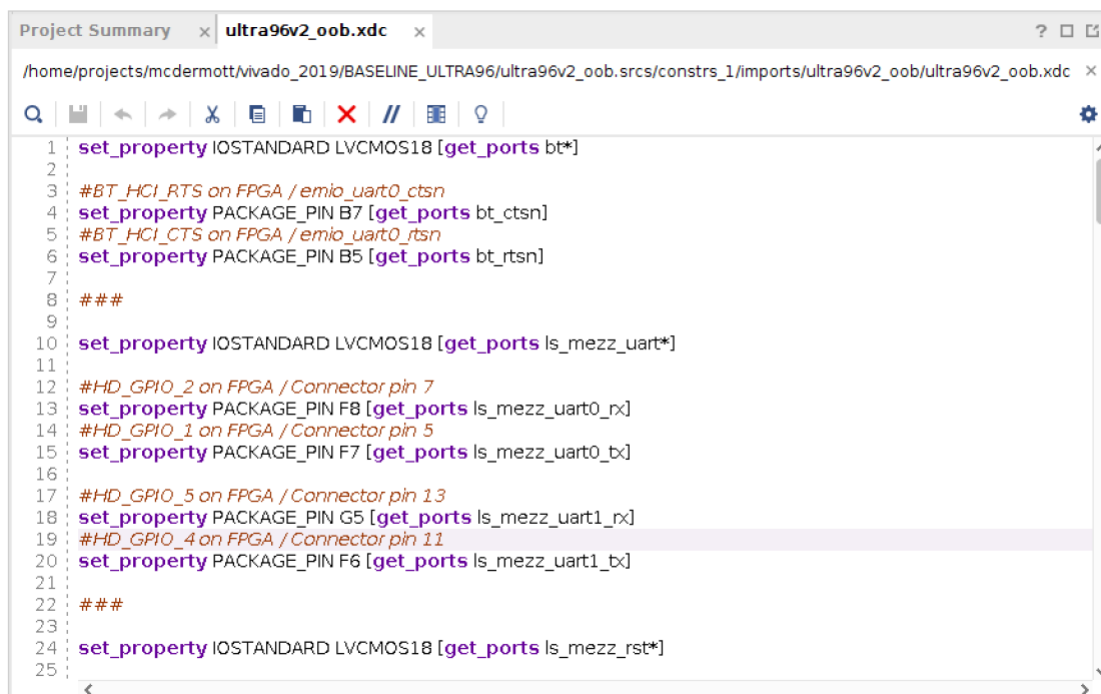
The AXI4_BURST_MASTER will be reconfigured in LAB #2 to include an interrupt output. All components in PL which must communicate with the PS will be done using interrupts. Polling or spin-locking will not be permitted.

Setting up the Ultra96 Baseline Xilinx Environment

To confirm that your Vivado environment is working correctly, you need to generate a bitstream from the Out-of-Box (OOB) design that you downloaded. The first step to confirm that you have a constraints file. This file tells the synthesis tool the timing and pin constraints for every I/O in your design. In the SOURCES window confirm that the `ultra96v2_oob.xdc` file exists.



You can edit the file by double-clicking the file name:

The image shows the Vivado editor window with the file `ultra96v2_oob.xdc` open. The file path is `/home/projects/mcdermott/vivado_2019/BASELINE_ULTRA96/ultra96v2_oob.srcs/constrs_1/imports/ultra96v2_oob/ultra96v2_oob.xdc`. The editor shows the following content:

```
1 set_property IOSTANDARD LVCMOS18 [get_ports bt*]
2
3 #BT_HCI_RTS on FPGA / emio_uart0_ctsn
4 set_property PACKAGE_PIN B7 [get_ports bt_ctsn]
5 #BT_HCI_CTS on FPGA / emio_uart0_rtsn
6 set_property PACKAGE_PIN B5 [get_ports bt_rtsn]
7
8 ###
9
10 set_property IOSTANDARD LVCMOS18 [get_ports ls_mezz_uart*]
11
12 #HD_GPIO_2 on FPGA / Connector pin 7
13 set_property PACKAGE_PIN F8 [get_ports ls_mezz_uart0_rx]
14 #HD_GPIO_1 on FPGA / Connector pin 5
15 set_property PACKAGE_PIN F7 [get_ports ls_mezz_uart0_tx]
16
17 #HD_GPIO_5 on FPGA / Connector pin 13
18 set_property PACKAGE_PIN G5 [get_ports ls_mezz_uart1_rx]
19 #HD_GPIO_4 on FPGA / Connector pin 11
20 set_property PACKAGE_PIN F6 [get_ports ls_mezz_uart1_tx]
21
22 ###
23
24 set_property IOSTANDARD LVCMOS18 [get_ports ls_mezz_rst*]
25
```

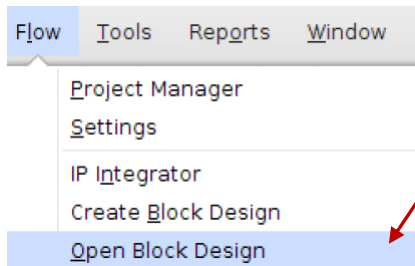
There are two SET_PROPERTY statements which are critical in this XDC file:

```
set_property IOSTANDARD LVCMOS18 [get_ports ls_mezz_uart*]
and
set_property PACKAGE_PIN F8 [get_ports ls_mezz_uart0_rx]
```

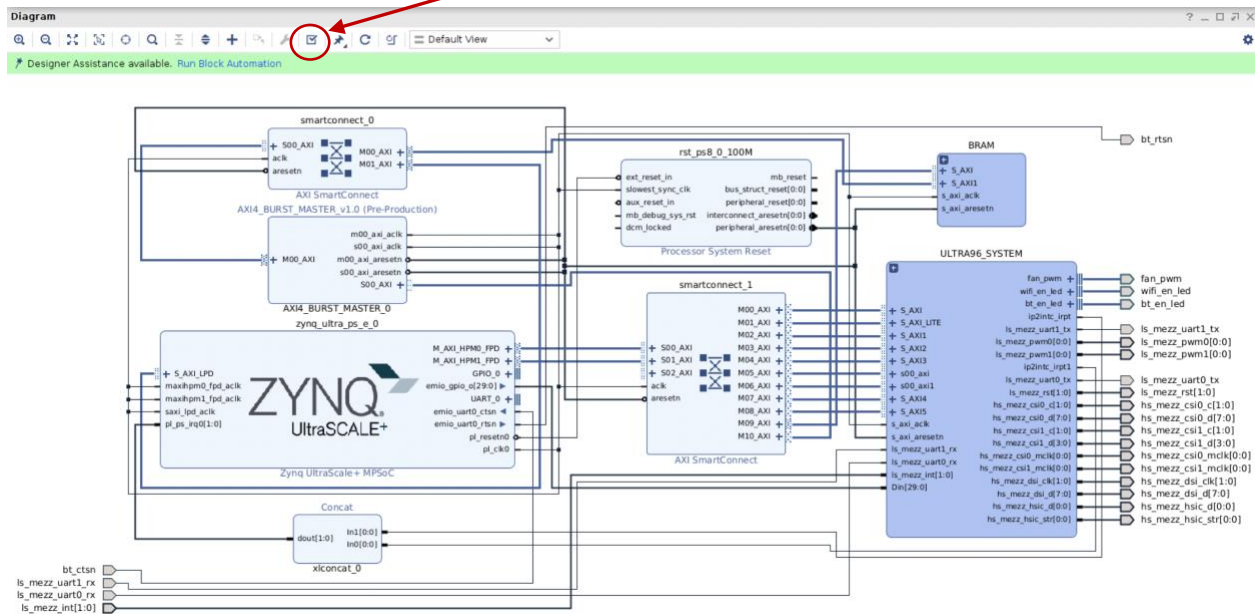
The first one sets the voltage drive level for the pin (or pins). The second assigns the signal to a particular pin on the FPGA. Both need to be present to generate a valid bitstream file.

Setting up the Ultra96 Baseline Xilinx Environment

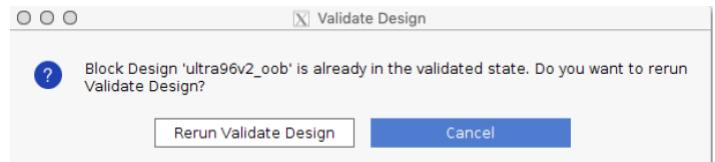
In the **Flow** dropdown select the **Open Block Design** command:



You will see the following schematic. Select the checkbox.



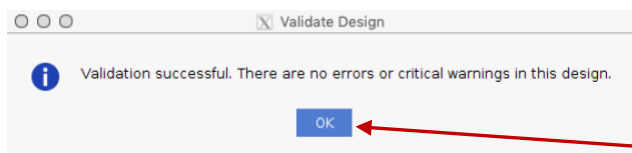
This may generate the following popup:



Select **Rerun Validate Design** This will confirm that the design is free of (most) errors.

NOTE: It is more of a linting tool...

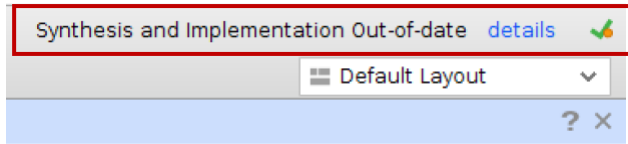
Once the command is completed the following popup will appear:



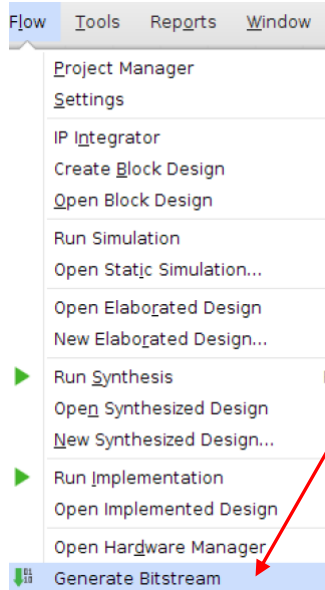
Click **OK**

Setting up the Ultra96 Baseline Xilinx Environment

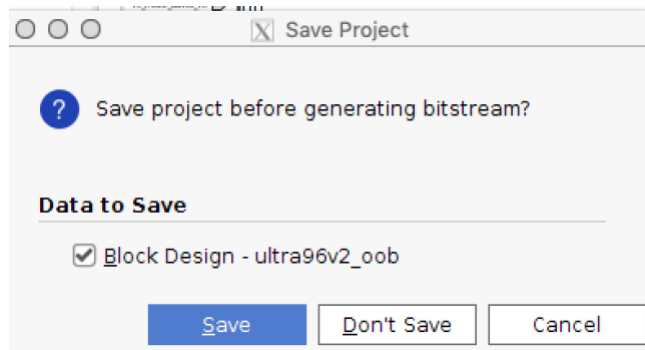
In the upper right-hand corner of the main Vivado window is the status of the design. Get use to looking at this status indicator. It is dynamic and will show what operations are executing.



In the **Flow** dropdown select the **Generate Bitstream** command:



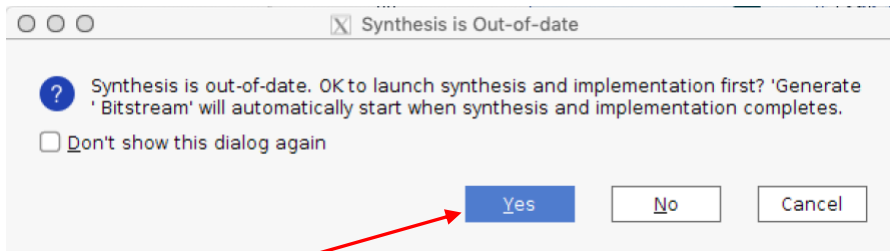
You should see the following popup:



Click **Save**

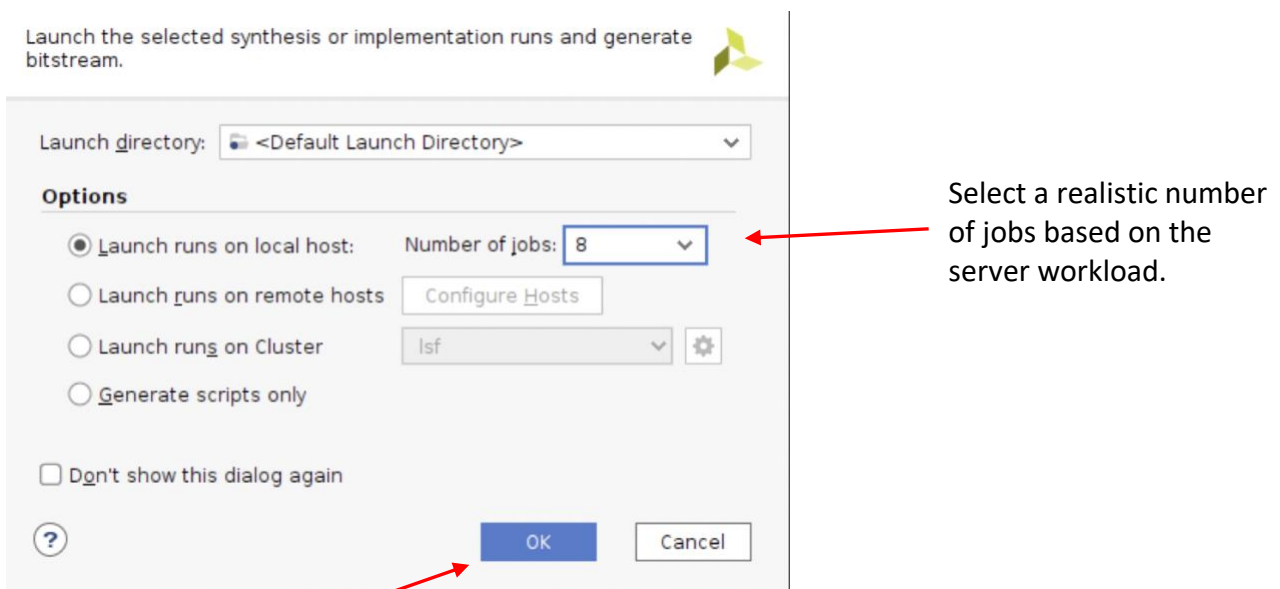
Setting up the Ultra96 Baseline Xilinx Environment

The following popup will appear:



Click **Yes**

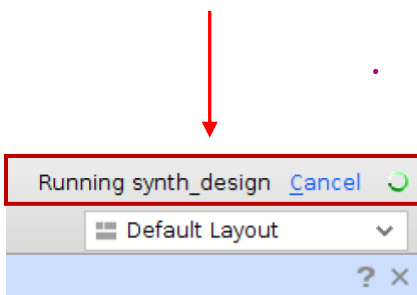
The following popup will appear:



Select a realistic number of jobs based on the server workload.

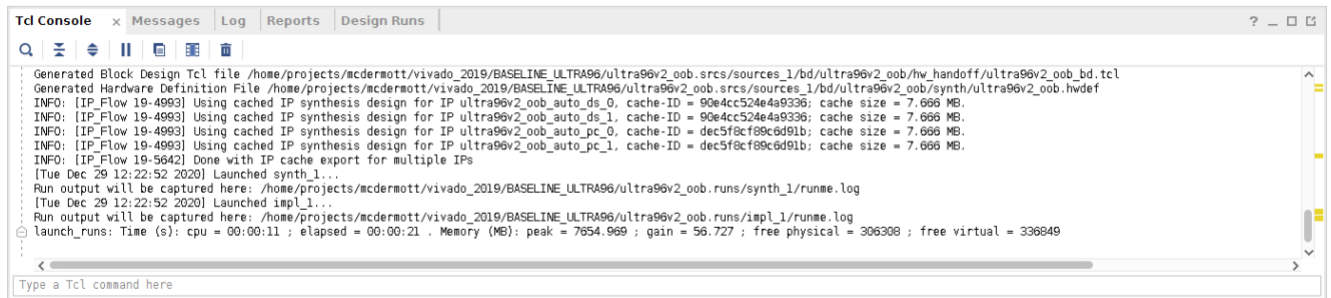
Click **OK**

The status indicator will show that the design is being synthesized:



Setting up the Ultra96 Baseline Xilinx Environment

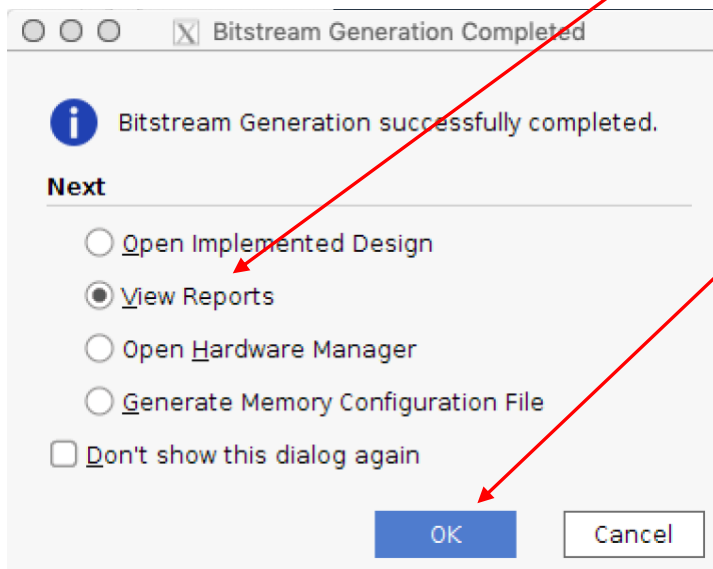
There is console window that will show the status of each operation:



```
Tcl Console x Messages Log Reports Design Runs
Generated Block Design Tcl file /home/projects/mcdermott/vivado_2019/BASELINE_ULTRA96/ultra96v2_oob.srcs/sources_1/bd/ultra96v2_oob/hw_handoff/ultra96v2_oob_bd.tcl
Generated Hardware Definition File /home/projects/mcdermott/vivado_2019/BASELINE_ULTRA96/ultra96v2_oob.srcs/sources_1/bd/ultra96v2_oob/synth/ultra96v2_oob.hwdef
INFO: [IP_Flow 19-4993] Using cached IP synthesis design for IP ultra96v2_oob_auto_ds_0, cache-ID = 90e4cc524e4a9336; cache size = 7.666 MB.
INFO: [IP_Flow 19-4993] Using cached IP synthesis design for IP ultra96v2_oob_auto_ds_1, cache-ID = 90e4cc524e4a9336; cache size = 7.666 MB.
INFO: [IP_Flow 19-4993] Using cached IP synthesis design for IP ultra96v2_oob_auto_pc_0, cache-ID = dec5f8cf69c6d91b; cache size = 7.666 MB.
INFO: [IP_Flow 19-4993] Using cached IP synthesis design for IP ultra96v2_oob_auto_pc_1, cache-ID = dec5f8cf69c6d91b; cache size = 7.666 MB.
INFO: [IP_Flow 19-5642] Done with IP cache export for multiple IPs
[Thu Dec 29 12:22:52 2020] Launched synth_1...
Run output will be captured here: /home/projects/mcdermott/vivado_2019/BASELINE_ULTRA96/ultra96v2_oob.runs/synth_1/runme.log
[Thu Dec 29 12:22:52 2020] Launched impl_1...
Run output will be captured here: /home/projects/mcdermott/vivado_2019/BASELINE_ULTRA96/ultra96v2_oob.runs/impl_1/runme.log
launch_runs: Time (s): cpu = 00:00:11 ; elapsed = 00:00:21 . Memory (MB): peak = 7654.969 ; gain = 56.727 ; free physical = 306308 ; free virtual = 336849
Type a Tcl command here
```

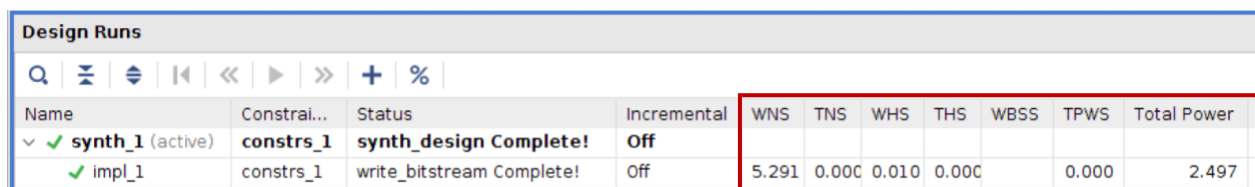
Pay special attention to errors and critical warnings.

Once the bitstream is generated select the **View Reports** button and click **OK**



There will be several reports that should be looked at if the design does not synthesize, place, or route correctly.

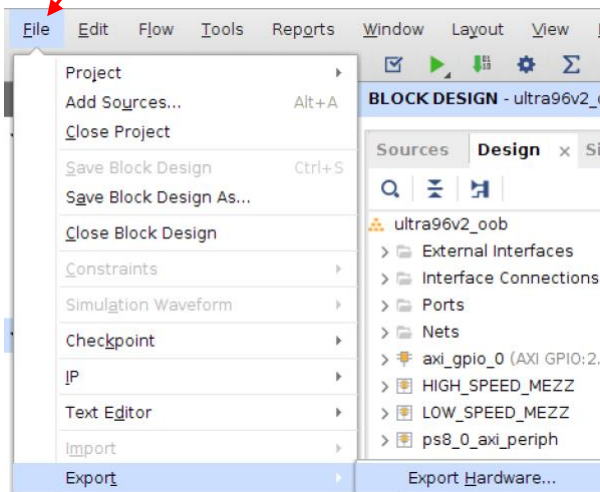
The key report to look at is under **Design Runs**. It will show the final STA (static timing analysis) runs for the design. The STA section highlighted below indicates that the design meets setup and hold time constraints:



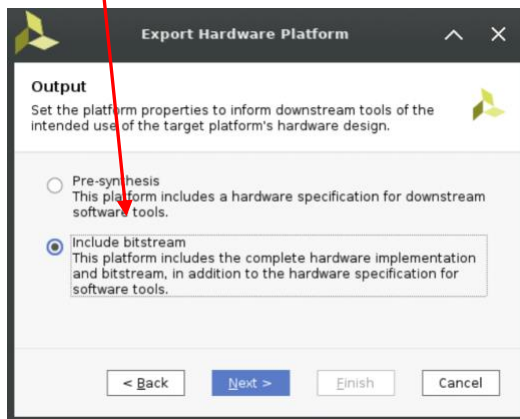
Design Runs										
Name	Constrai...	Status	Incremental	WNS	TNS	WHS	THS	WBSS	TPWS	Total Power
✓ synth_1 (active)	constrs_1	synth_design Complete!	Off							
✓ impl_1	constrs_1	write_bitstream Complete!	Off	5.291	0.000	0.010	0.000		0.000	2.497

Setting up the Ultra96 Baseline Xilinx Environment

Under **File** select **Export** and then **Export Hardware**:



Select **Include Bitstream** and click **Next**



The bit file is in the following sub-directory:

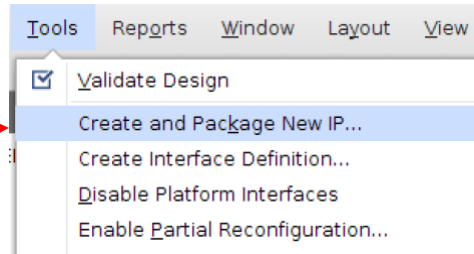
`../ultra96v2_oob.runs/impl_1/ultra96v2_oob_wrapper.bit`

This bit file can now be loaded into the FPGA using the [fpgautil](#) program. More about that in the Lab 1 writeup.

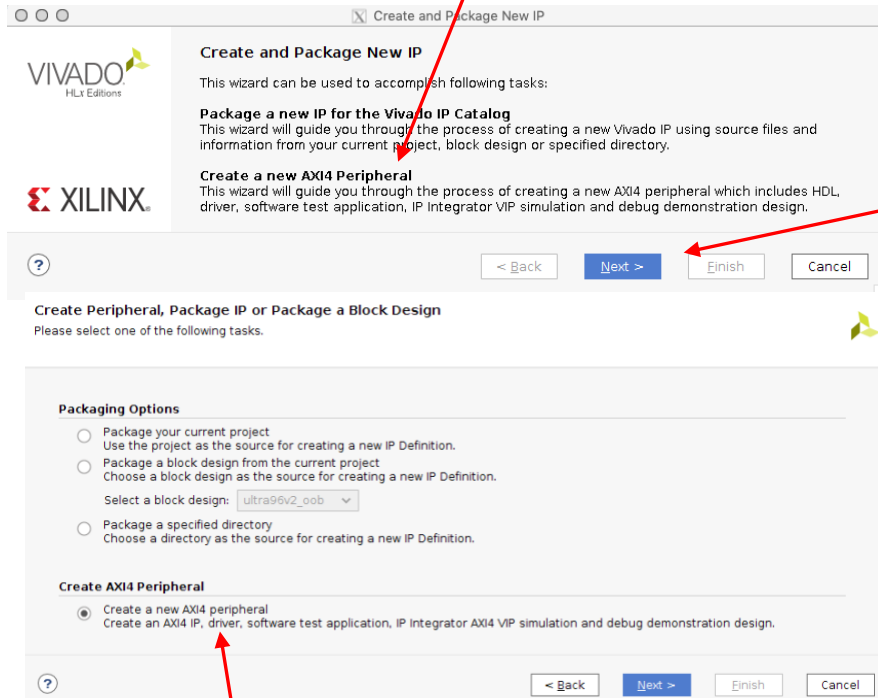
Setting up the Ultra96 Baseline Xilinx Environment

Generating New IP

Now we are going to generate new IP for use in the OOB schematic. Under the **Tools** pulldown select **Create and Package New IP**:



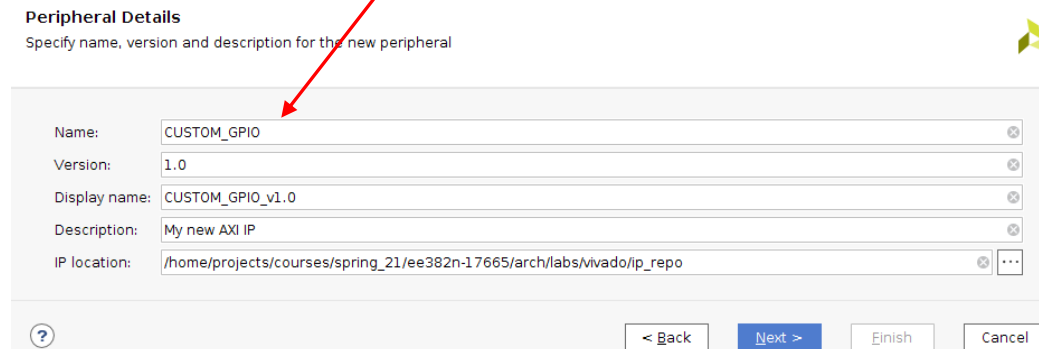
We will be creating a **New AXI4 Peripheral** that will be attached to the AXI Crossbar.



Select **NEXT**

Select **Create AXI4 Peripheral** and click **Next**.

Under **Name** enter **CUSTOM_GPIO**

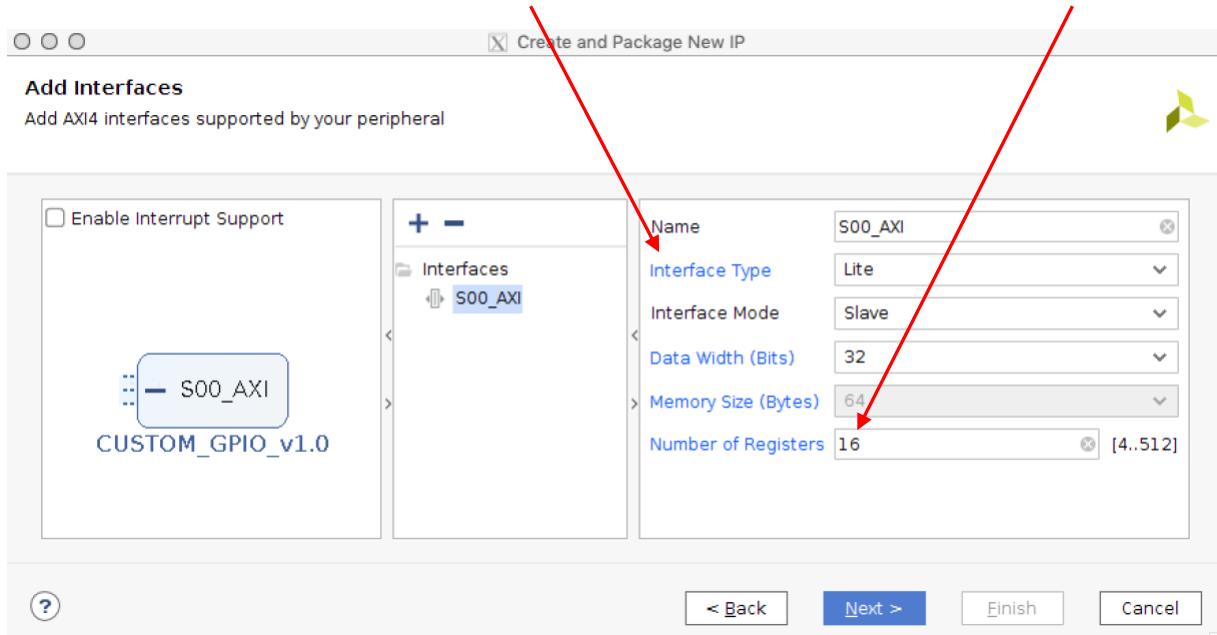


Select a location for the IP Repository. Generally, it should be in your Vivado ip_repo directory:
~/vivado/ip_repo/

Click **Next**

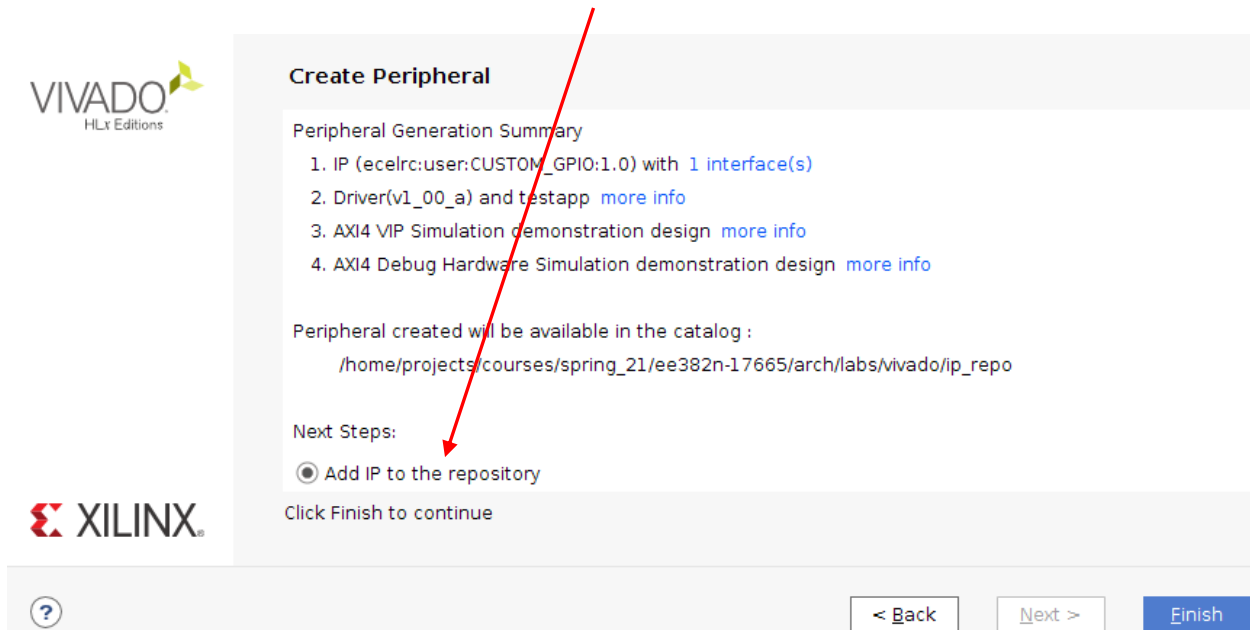
Setting up the Ultra96 Baseline Xilinx Environment

In the following popup select **Interface Type = Lite** and **Number of Registers = 16**



Click **NEXT**.

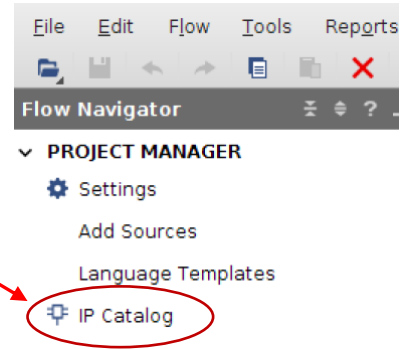
In the popup window select **ADD IP to the Repository**



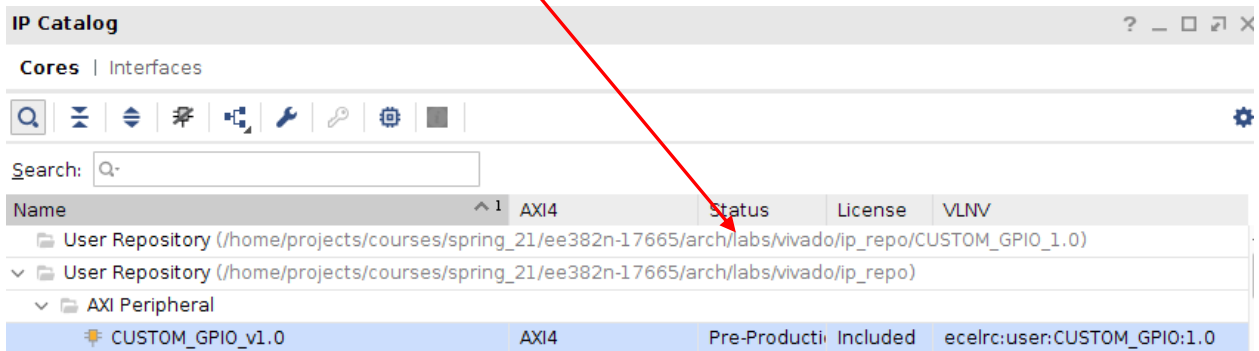
Click **Finish**

Setting up the Ultra96 Baseline Xilinx Environment

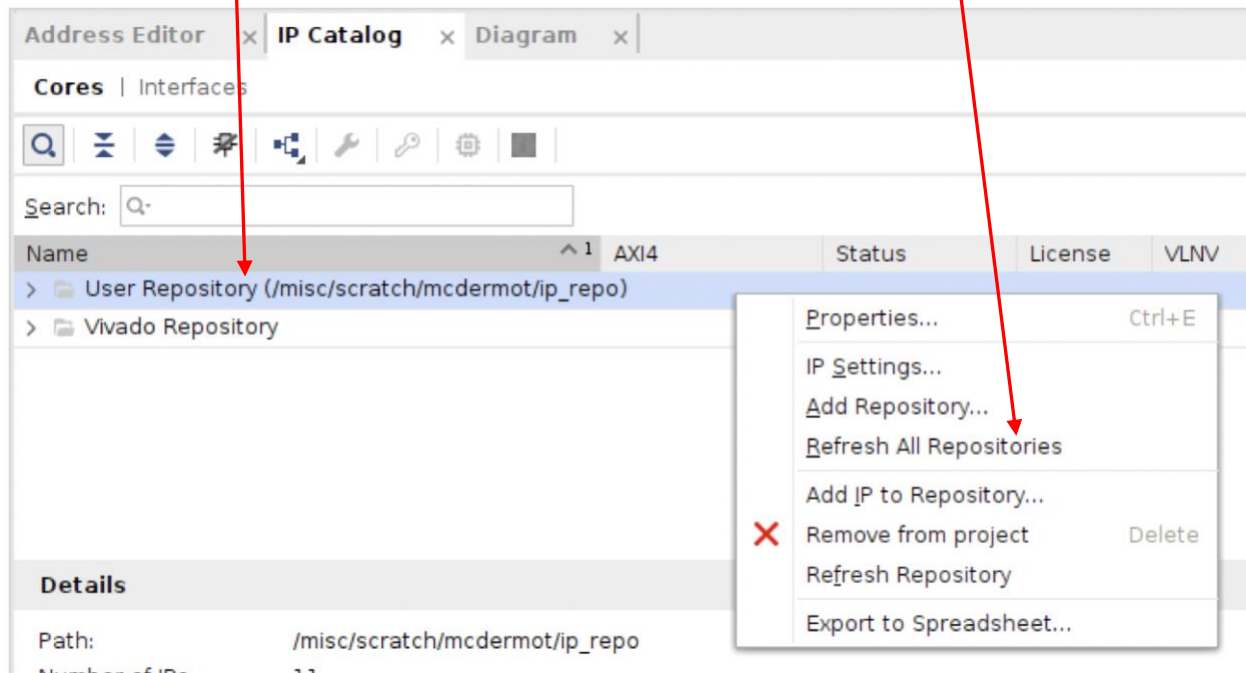
The next step is to edit the IP and add I/O ports. Under **Flow Navigator** select **IP Catalog**



There is a bug in the Xilinx SW. When you add a new component to the IP REPO you may see two user repositories show up in the IP Catalog. Using the right mouse key and remove the following repository from the project.



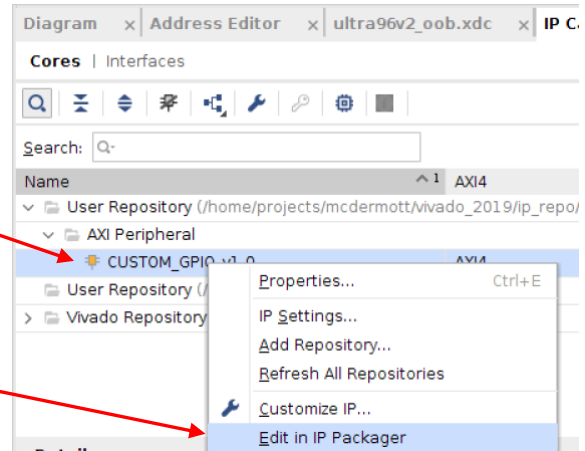
Now select the main IP REPO, using the right mouse key select Refresh All Repositories



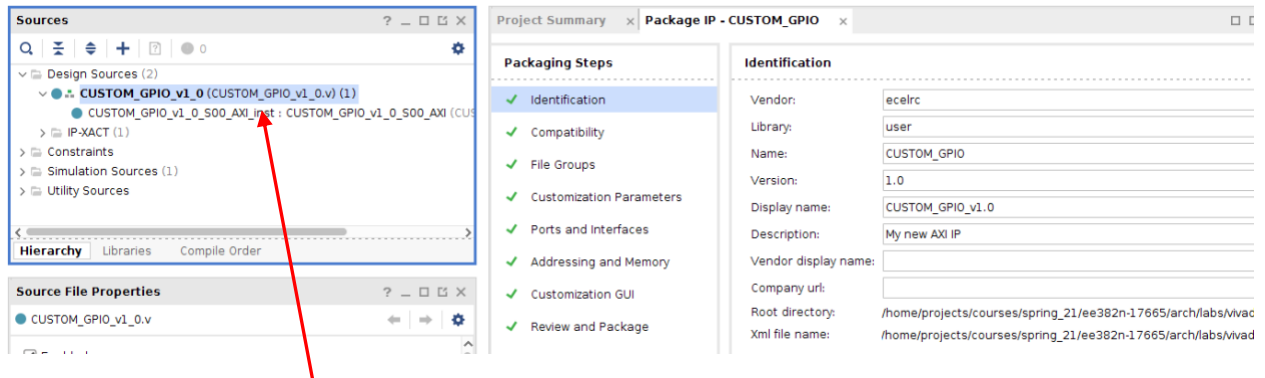
Setting up the Ultra96 Baseline Xilinx Environment

In the **IP Catalog** window Select **CUSTOM_GPIO_v1.0**

Using the right mouse key select **Edit in IP Packager**

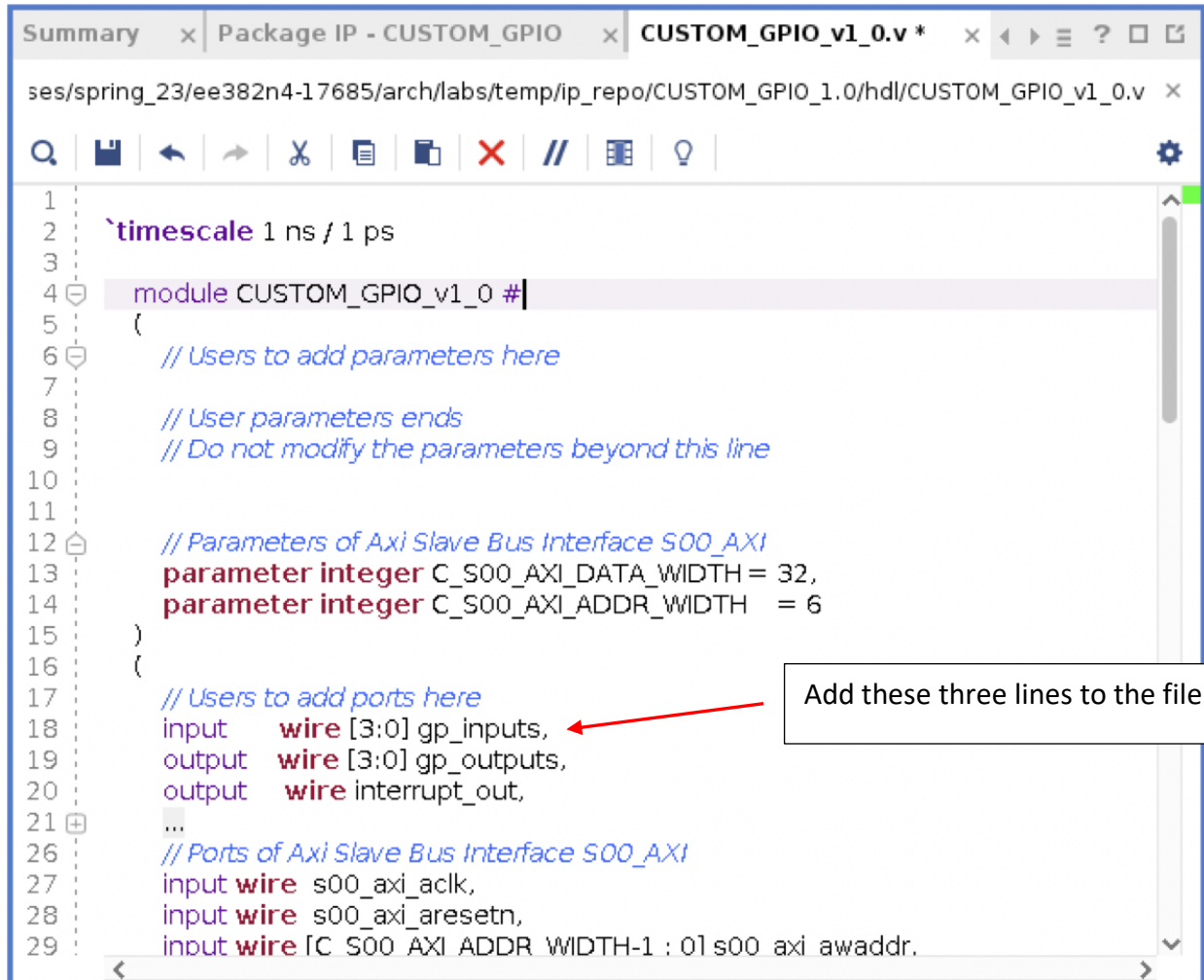


This will create a new project window. The **Sources** window is used to edit the **CUSTOM_GPIO** Verilog code. The **Package IP** window is used to repackage the IP once the Verilog has been edited.



Double click **CUSTOM_GPIO_v1_0** The file will open in text editing window as shown on the next page.

Setting up the Ultra96 Baseline Xilinx Environment



```
1 |  
2 | `timescale 1 ns / 1 ps  
3 |  
4 | module CUSTOM_GPIO_v1_0 #  
5 | (  
6 |     // Users to add parameters here  
7 |  
8 |     // User parameters ends  
9 |     // Do not modify the parameters beyond this line  
10 |  
11 |  
12 |     // Parameters of Axi Slave Bus Interface S00_AXI  
13 |     parameter integer C_S00_AXI_DATA_WIDTH = 32,  
14 |     parameter integer C_S00_AXI_ADDR_WIDTH = 6  
15 | )  
16 | (  
17 |     // Users to add ports here  
18 |     input  wire [3:0] gp_inputs, ←  
19 |     output wire [3:0] gp_outputs,  
20 |     output wire interrupt_out,  
21 |     ...  
26 |     // Ports of Axi Slave Bus Interface S00_AXI  
27 |     input wire s00_axi_aclk,  
28 |     input wire s00_axi_aresetn,  
29 |     input wire [C_S00_AXI_ADDR_WIDTH-1 : 0] s00_axi_awaddr,
```

Add these three lines to the file

```
49 | // Instantiation of Axi Bus Interface S00_AXI  
50 | CUSTOM_GPIO_v1_0_S00_AXI # (  
51 |     .C_S_AXI_DATA_WIDTH(C_S00_AXI_DATA_WIDTH),  
52 |     .C_S_AXI_ADDR_WIDTH(C_S00_AXI_ADDR_WIDTH)  
53 | ) CUSTOM_GPIO_v1_0_S00_AXI_inst (  
54 |     .gp_inputs(gp_inputs),  
55 |     .gp_outputs(gp_outputs), ←  
56 |     .interrupt_out(interrupt_out),
```

And add these three lines to the file

Setting up the Ultra96 Baseline Xilinx Environment



We need add some Verilog code to the next level of hierarchy. Double-click the highlighted file

```
17 // Users to add ports here
18 input wire [3:0] gp_inputs,
19 output wire [3:0] gp_outputs,
20 output wire interrupt_out,
```

Add these three lines to the file

```
494 // Address decoding for reading registers
495 case ( axi_araddr[ADDR_LSB+OPT_MEM_ADDR_BITS:ADDR_LSB] )
496     4'h0 : reg_data_out <= slv_reg0;
497     //4'h1 : reg_data_out <= slv_reg1;
498     4'h1 : reg_data_out <= {27'b0, gp_inputs};
```

Comment out this line

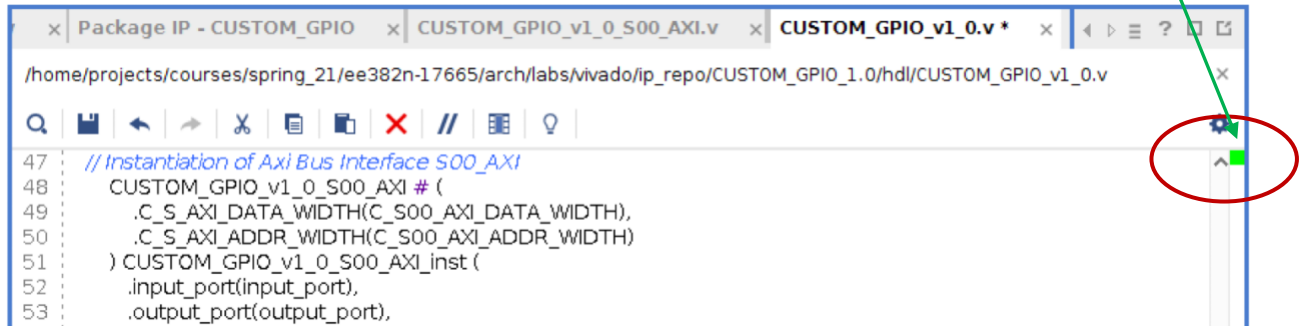
Add this line.

```
536 // Add user logic here
537 assign gp_outputs[3:0] = slv_reg0[3:0];
538 assign interrupt_out = slv_reg2[0];
539
540 // User logic ends
```

Add these two lines.

Setting up the Ultra96 Baseline Xilinx Environment

The built-in Xilinx editor does dynamic syntax checking. Get in the habit of looking at the **green** indicator before saving your files



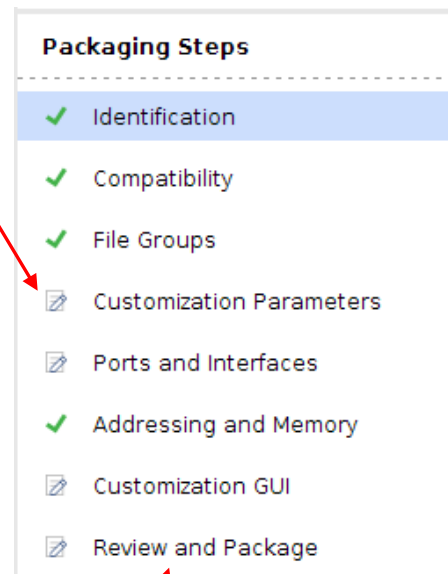
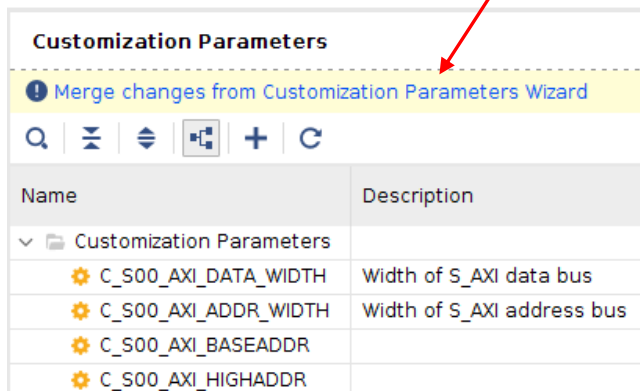
This will save a bunch of time because you won't know you have a problem until you get into the synthesis stage.

Save both files. We are now going to package up the IP in the Repository.

Go to the Package IP window and click

Customization Parameters

Next, click **Merge Changes Customization Parameters**



And then click **Review and Package**

Setting up the Ultra96 Baseline Xilinx Environment

The following popup will appear.

Review and Package

IP has been modified.

Summary

Display name: CUSTOM_GPIO_v1.0
Description: My new AXI IP
Root directory: /home/projects/courses/spring_21/ee382n-17665/arch/la

Click **Re-Package IP**

After Packaging

An archive will not be generated. Use the settings link below to change
Project will be removed after completion
[Edit packaging settings](#)

Re-Package IP

Click **YES**

Finished packaging 'CUSTOM_GPIO_v1.0' successfully.
Package IP Location: /home/projects/courses/spring_21/ee382n-17665/arch/labs/vivado/ip_repo/CUSTOM_GPIO_1.0

Do you want to close the project?

Yes No

Generally, when you save IP you will be asked to upgrade the schematic. Click on

Show IP Status

BLOCKDESIGN - ultra96v2_oob

/CUSTOM_GPIO_0 block in this design should be upgraded. Show IP Status Upgrade Later

Sources Design x Signals Board Platfo ? _ □ □

ultra96v2_oob

- External Interfaces
- Interface Connections
- Ports
- Nets
- axi_gpio_0 (AXI GPIO:2.0)
- CUSTOM_GPIO_0 (CUSTOM_GPIO_v1.0:1.0)**
- High_Speed_MEZZ
- Low_Speed_MEZZ

Block Properties

CUSTOM_GPIO_0

Name: CUSTOM_GPIO_0

Parent name: ultra96v2_oob

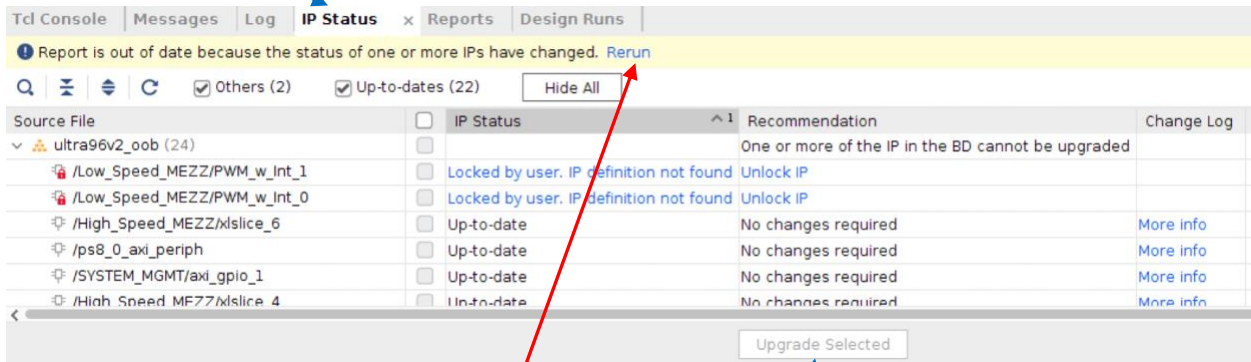
General Properties IP

Address Editor

Cell	Slave Interface	Slave Segment	Offset Address
zynq_ultra_ps_e_0			
Data (40 address bits : 0x00A0000000 [256M], 0x0400000000 [4G], 0x1000000000 [224G], 0x00B0000000 [256M], 0x0500000000 [4G])			
Low_Speed_MEZZ/PWM_w_int_0	s00_axi	reg0	0x00_A002_3000
Low_Speed_MEZZ/PWM_w_int_1	s00_axi	reg0	0x00_A002_4000
axi_gpio_0	S_AXI	Reg	0x00_A002_0000
SYSTEM_MGMT/axi_gpio_1	S_AXI	Reg	0x00_A002_1000
Low_Speed_MEZZ/axi_gpio_2	S_AXI	Reg	0x00_A002_2000
SYSTEM_MGMT/axi_gpio_3	S_AXI	Reg	0x00_A002_5000
Low_Speed_MEZZ/axi_uart16550_0	S_AXI	Reg	0x00_A000_0000
Low_Speed_MEZZ/axi_uart16550_1	S_AXI	Reg	0x00_A001_0000
SYSTEM_MGMT/system_management_wiz_0	S_AXI_LITE	Reg	0x00_A002_6000
Unconnected Slaves			
CUSTOM_GPIO_0	S00_AXI	S00_AXI_reg	

Setting up the Ultra96 Baseline Xilinx Environment

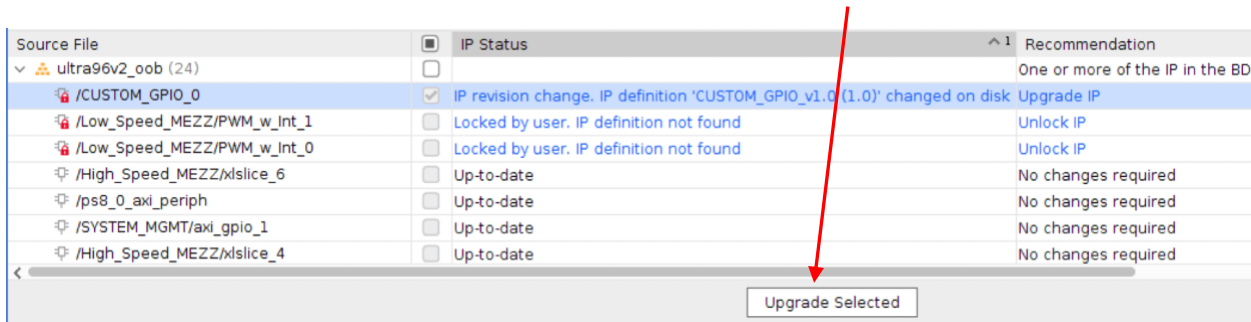
The **IP Status** window will be shown in the following window.



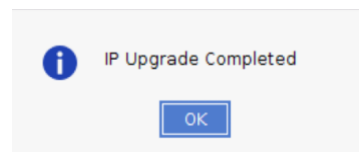
then click on **Rerun**

If Upgrade Selected is grayed out

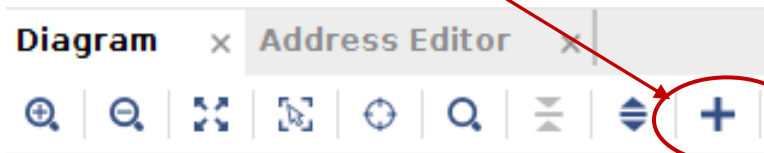
Then click on **Upgrade Selected**



Click **OK** when the IP Upgrade is Completed

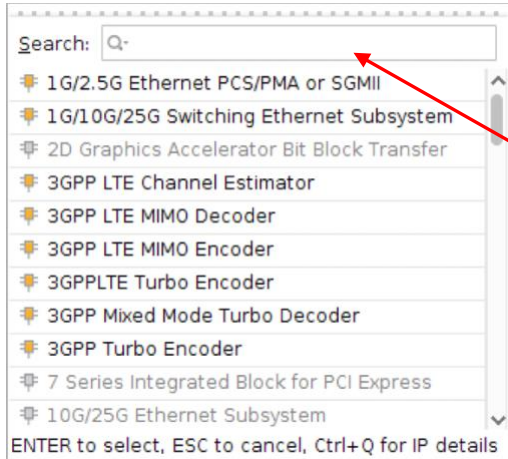


Let's instantiate the **CUSTOM_GPIO_v1_0** block into the MAIN schematic. In the Diagram window click the **plus** sign



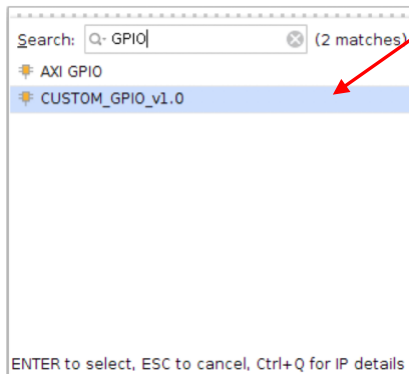
Setting up the Ultra96 Baseline Xilinx Environment

The following popup will appear.

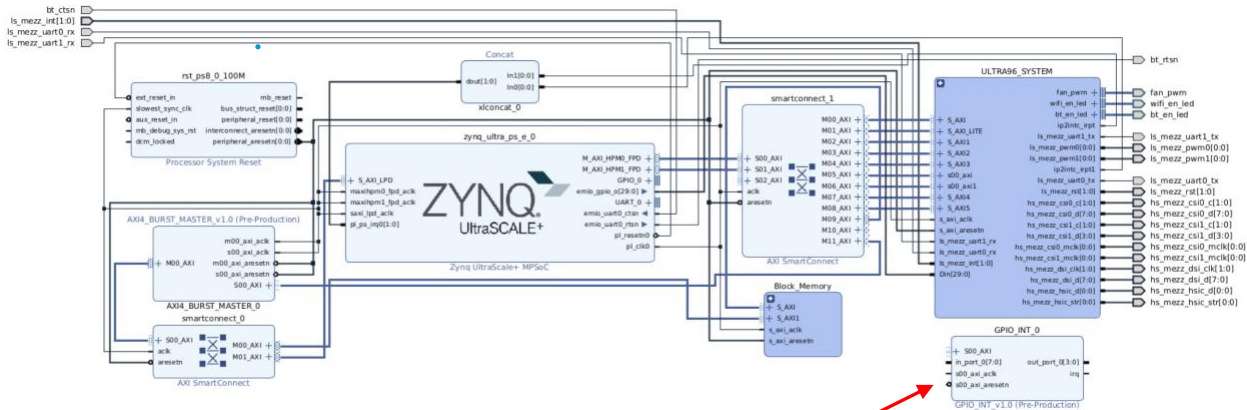


Type GPIO in the search window. You will see the block you just built as well as one provided by Xilinx.

Double Click **CUSTOM_GPIO_v1_0**



Setting up the Ultra96 Baseline Xilinx Environment

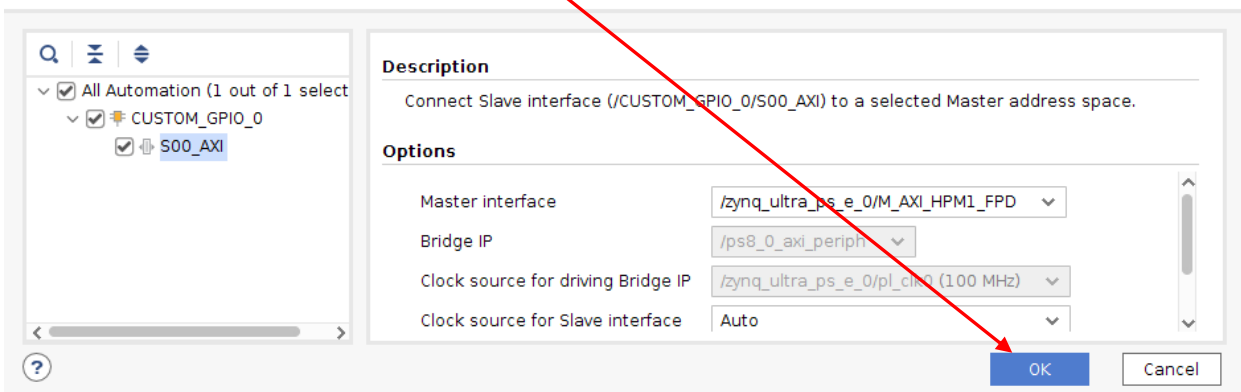


Once the **CUSTOM_GPIO_v1_0** block has been instantiated

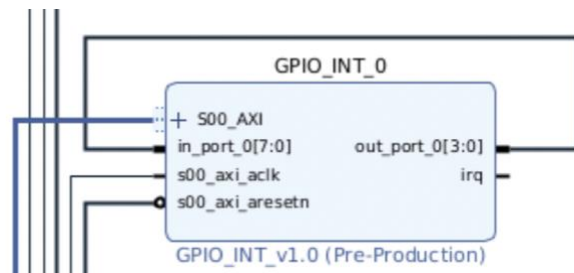
click on **Run Connection Automation**

Click **OK** in the following popup:

Automatically make connections in your design by checking the boxes of the interfaces to connect. Select an interface on the left to display its configuration options on the right.

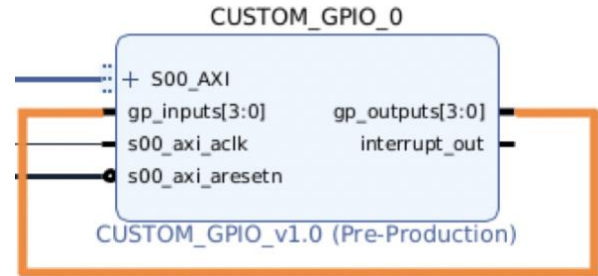


The **CUSTOM_GPIO_v1_0** block will be automatically hooked up the crossbar switch

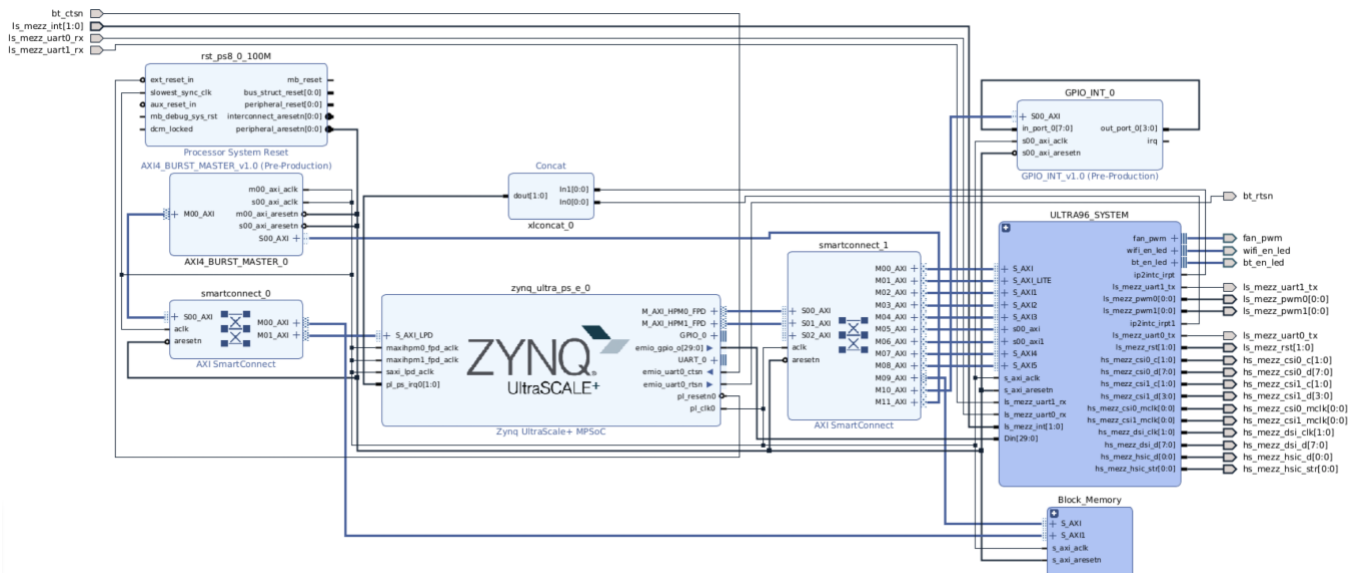


Setting up the Ultra96 Baseline Xilinx Environment

Now we need to hook up the outputs to the inputs on the **CUSTOM_GPIO_v1_0** block



The final schematic will look something like this:



Setting up the Ultra96 Baseline Xilinx Environment

Double check that an address has been assigned to the **CUSTOM_GPIO** block

Name	Interface	Slave Segment	Master Base Address	Range	Master High Address
AXI4_BURST_MASTER_0					
/AXI4_BURST_MASTER_0/M00_AXI (40 address bits : 1T)					
/Block_Memory/axi_bram_ctrl_1/S_AXI	S_AXI	Mem0	0x00_C000_0000	8K	0x00_C000_1FFF
/zynq_ultra_ps_e_0/SAXIGP6	S_AXI_LPD	LPD_LPS_OCM	0x00_FF80_0000	8M	0x00_FFFF_FFFF
/zynq_ultra_ps_e_0/SAXIGP6	S_AXI_LPD	.PD_DDR_LOW	0x00_4000_0000	1G	0x00_7FFF_FFFF
zynq_ultra_ps_e_0					
/zynq_ultra_ps_e_0/Data					
/AXI4_BURST_MASTER_0/S00_AXI	S00_AXI	S00_AXI_reg	0x00_A000_0000	4K	0x00_A000_0FFF
/AXI4_BURST_MASTER_0/S00_AXI	S00_AXI	S00_AXI_reg	0x00_A000_0000	4K	0x00_A000_0FFF
/Block_Memory/axi_bram_ctrl_0/S_AXI	S_AXI	Mem0	0x00_A000_2000	8K	0x00_A000_3FFF
/Block_Memory/axi_bram_ctrl_0/S_AXI	S_AXI	Mem0	0x00_A000_2000	8K	0x00_A000_3FFF
/ULTRA96_SYSTEM/BD_CTL_GPIO/axi_gpio_0/S_AXI	S_AXI	Reg	0x00_A001_0000	64K	0x00_A001_FFFF
/ULTRA96_SYSTEM/BD_CTL_GPIO/axi_gpio_0/S_AXI	S_AXI	Reg	0x00_A001_0000	64K	0x00_A001_FFFF
/ULTRA96_SYSTEM/BD_CTL_GPIO/axi_gpio_1/S_AXI	S_AXI	Reg	0x00_A002_0000	64K	0x00_A002_FFFF
/ULTRA96_SYSTEM/BD_CTL_GPIO/axi_gpio_1/S_AXI	S_AXI	Reg	0x00_A002_0000	64K	0x00_A002_FFFF
/ULTRA96_SYSTEM/Low_Speed_MEZZ/PWM_w_Int_0/s00_axi	s00_axi	reg0	0x00_A003_0000	64K	0x00_A003_FFFF
/ULTRA96_SYSTEM/Low_Speed_MEZZ/PWM_w_Int_0/s00_axi	s00_axi	reg0	0x00_A003_0000	64K	0x00_A003_FFFF
/ULTRA96_SYSTEM/Low_Speed_MEZZ/PWM_w_Int_1/s00_axi	s00_axi	reg0	0x00_A004_0000	64K	0x00_A004_FFFF
/ULTRA96_SYSTEM/Low_Speed_MEZZ/PWM_w_Int_1/s00_axi	s00_axi	reg0	0x00_A004_0000	64K	0x00_A004_FFFF
/ULTRA96_SYSTEM/Low_Speed_MEZZ/axi_gpio_2/S_AXI	S_AXI	Reg	0x00_A005_0000	64K	0x00_A005_FFFF
/ULTRA96_SYSTEM/Low_Speed_MEZZ/axi_gpio_2/S_AXI	S_AXI	Reg	0x00_A005_0000	64K	0x00_A005_FFFF
/ULTRA96_SYSTEM/Low_Speed_MEZZ/axi_uart16550_0/S_AXI	S_AXI	Reg	0x00_A006_0000	64K	0x00_A006_FFFF
/ULTRA96_SYSTEM/Low_Speed_MEZZ/axi_uart16550_0/S_AXI	S_AXI	Reg	0x00_A006_0000	64K	0x00_A006_FFFF
/ULTRA96_SYSTEM/Low_Speed_MEZZ/axi_uart16550_1/S_AXI	S_AXI	Reg	0x00_A007_0000	64K	0x00_A007_FFFF
/ULTRA96_SYSTEM/Low_Speed_MEZZ/axi_uart16550_1/S_AXI	S_AXI	Reg	0x00_A007_0000	64K	0x00_A007_FFFF
/ULTRA96_SYSTEM/SYS_MGMT/axi_gpio_3/S_AXI	S_AXI	Reg	0x00_A008_0000	64K	0x00_A008_FFFF
/ULTRA96_SYSTEM/SYS_MGMT/axi_gpio_3/S_AXI	S_AXI	Reg	0x00_A008_0000	64K	0x00_A008_FFFF
/ULTRA96_SYSTEM/SYS_MGMT/system_management_wiz_0/S_AXI_LITE	S_AXI_LITE	Reg	0x00_A009_0000	64K	0x00_A009_FFFF
/ULTRA96_SYSTEM/SYS_MGMT/system_management_wiz_0/S_AXI_LITE	S_AXI_LITE	Reg	0x00_A009_0000	64K	0x00_A009_FFFF
/CUSTOM_GPIO_0/S00_AXI	S_AXI_LITE	Reg	0x00_A00A_0000	64K	0x00_A00A_FFFF

Make sure that your address map matches the one above. The Device Tree Blob (DTB) will be generated from this mapping (more about this later in the semester)