



# Professional PYNQ™ Lab

Adam Taylor

[Adam@AdiuvoEngineering.com](mailto:Adam@AdiuvoEngineering.com)

# Objective

The objectives of this Lab are:

1. Treat the AMD FFT IP core as the unit under test (UUT)
2. Demonstrate how to create an Overlay to test the UUT
3. Demonstrate different methods of control from PS and the impacts possible on performance.
4. Demonstrate how to display performance of the UUT in Jupyter labs

# Lab: FFT Verification

Open a browser and go to

[www.pynq.io](http://www.pynq.io)



[Home](#) [Get Started](#) [Boards](#) [Community](#) [Source Code](#) [Support](#)

## What is PYNQ?

PYNQ is an open-source project from AMD® that makes it easier to use *Adaptive Computing* platforms. Using the Python language and libraries, designers can exploit the benefits of programmable logic and microprocessors to build more capable and exciting electronic systems. PYNQ can be used with *Zynq*, *Zynq UltraScale+*, *Zynq RFSoc*, *Alveo* accelerator boards and AWS-F1 to create high performance applications with:

- parallel hardware execution
- high frame-rate video processing
- hardware accelerated algorithms
- real-time signal processing
- high bandwidth IO
- low latency control

**AMD**  
**PYNQ**



# Lab: FFT Verification

Select the boards page and  
download the SD card  
image for ZU1 CG

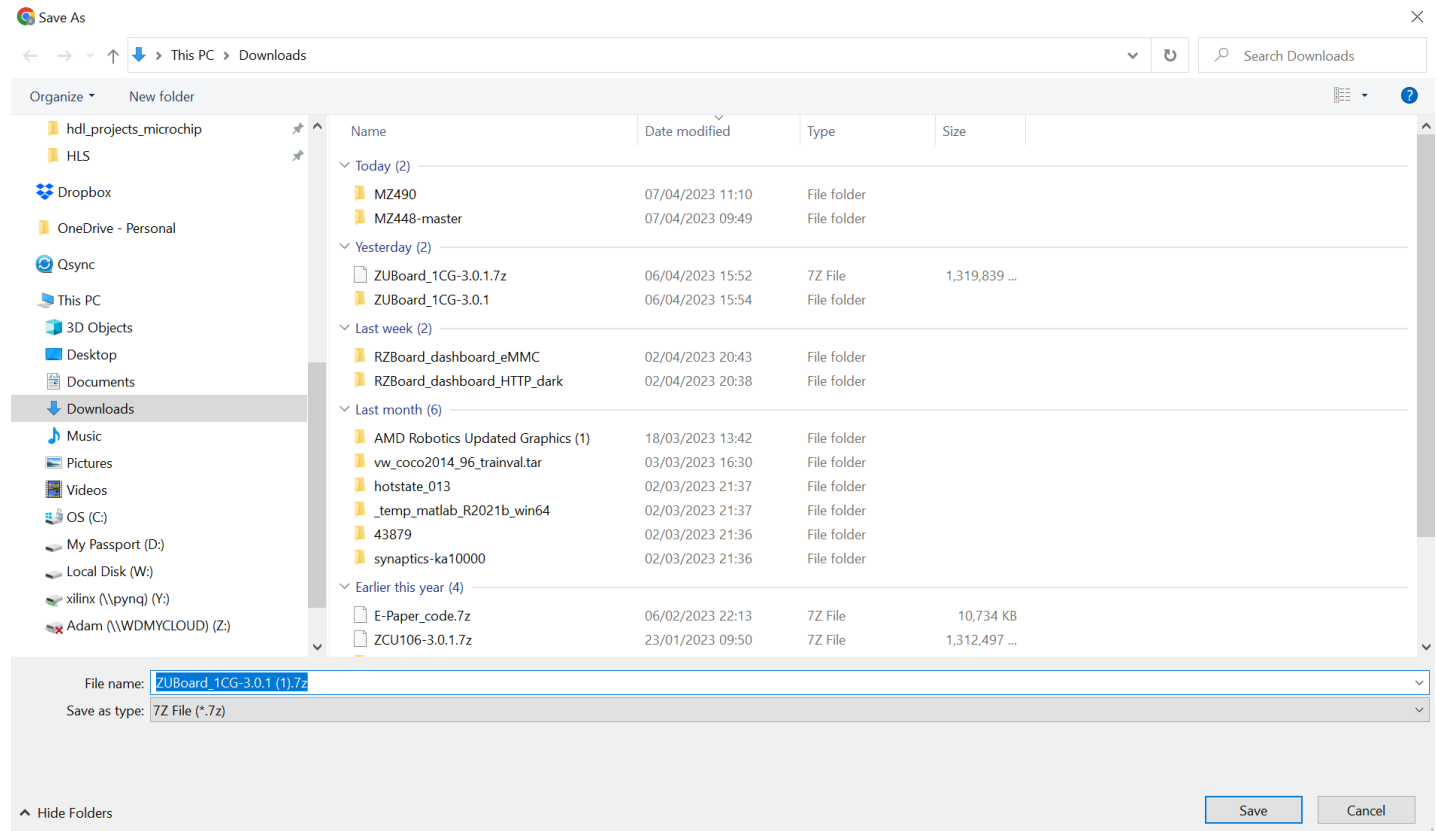
## Downloadable PYNQ images

If you have a Zynq board, you need a PYNQ SD card image to get started. You can download a pre-compiled PYNQ image from the table below. If an image is not available for your board, you can build your own SD card image (see details below).

Board	SD card image	Previous versions	Documentation	Board webpage
PYNQ-Z2	<a href="#">v3.0.1</a>	<a href="#">v2.7</a> <a href="#">v2.6</a>	<a href="#">PYNQ setup guide</a>	<a href="#">TUL Pynq-Z2</a>
PYNQ-Z1	<a href="#">v3.0.1</a>	<a href="#">v2.7</a> <a href="#">v2.6</a>	<a href="#">PYNQ setup guide</a>	<a href="#">Digilent Pynq-Z1</a>
PYNQ-ZU	<a href="#">v3.0.1</a>	<a href="#">v2.7</a> <a href="#">v2.6</a>	<a href="#">GitHub project page</a>	<a href="#">TUL PYNQ-ZU</a>
Kria KV260*	<a href="#">Ubuntu 22.04</a>		<a href="#">Kria PYNQ setup</a>	<a href="#">Xilinx Kria KV260</a>
Kria KR260*	<a href="#">Ubuntu 22.04</a>		<a href="#">Kria PYNQ setup</a>	<a href="#">Xilinx Kria KR260</a>
ZCU104	<a href="#">v3.0.1</a>	<a href="#">v2.7</a> <a href="#">v2.6</a>	<a href="#">PYNQ setup guide</a>	<a href="#">Xilinx ZCU104</a>
RFSoc 2x2	<a href="#">v3.0.1</a>	<a href="#">v2.7</a> <a href="#">v2.6</a>	<a href="#">RFSoc-PYNQ</a>	<a href="#">XUP RFSoc 2x2</a>
RFSoc 4x2	<a href="#">v3.0.1</a>	<a href="#">v2.7</a>	<a href="#">RFSoc-PYNQ</a>	<a href="#">XUP RFSoc 4x2</a>
ZCU111	<a href="#">v3.0.1</a>	<a href="#">v2.7</a> <a href="#">v2.6</a>	<a href="#">RFSoc-PYNQ</a>	<a href="#">Xilinx ZCU111</a>
ZCU208	<a href="#">v3.0.1</a>		<a href="#">RFSoc-PYNQ</a>	<a href="#">Xilinx ZCU208</a>
Ultra96V2	<a href="#">v3.0.1</a>	<a href="#">v2.7</a> <a href="#">v2.6</a>	<a href="#">Avnet PYNQ webpage</a>	<a href="#">Avnet Ultra96V2</a>
Ultra96 (legacy)	<a href="#">v3.0.1</a>	<a href="#">v2.7</a> <a href="#">v2.6</a>	See Ultra96V2	See Ultra96V2
ZUBoard 1CG	<a href="#">v3.0.1</a>		<a href="#">GitHub project page</a>	<a href="#">Avnet ZUBoard 1CG</a>

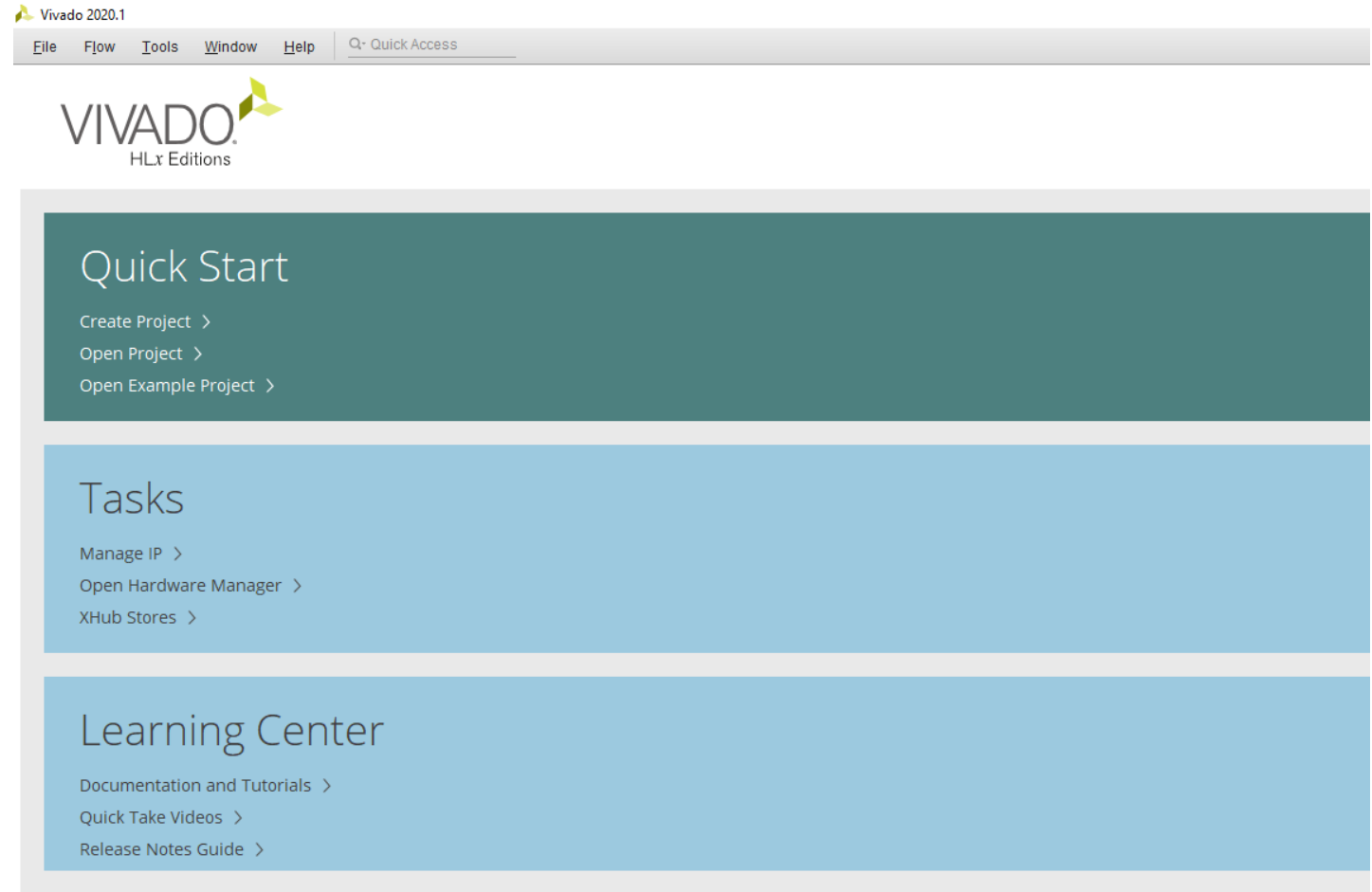
# Lab: FFT Verification

Save the SD Card image to a preferred location on your local computer



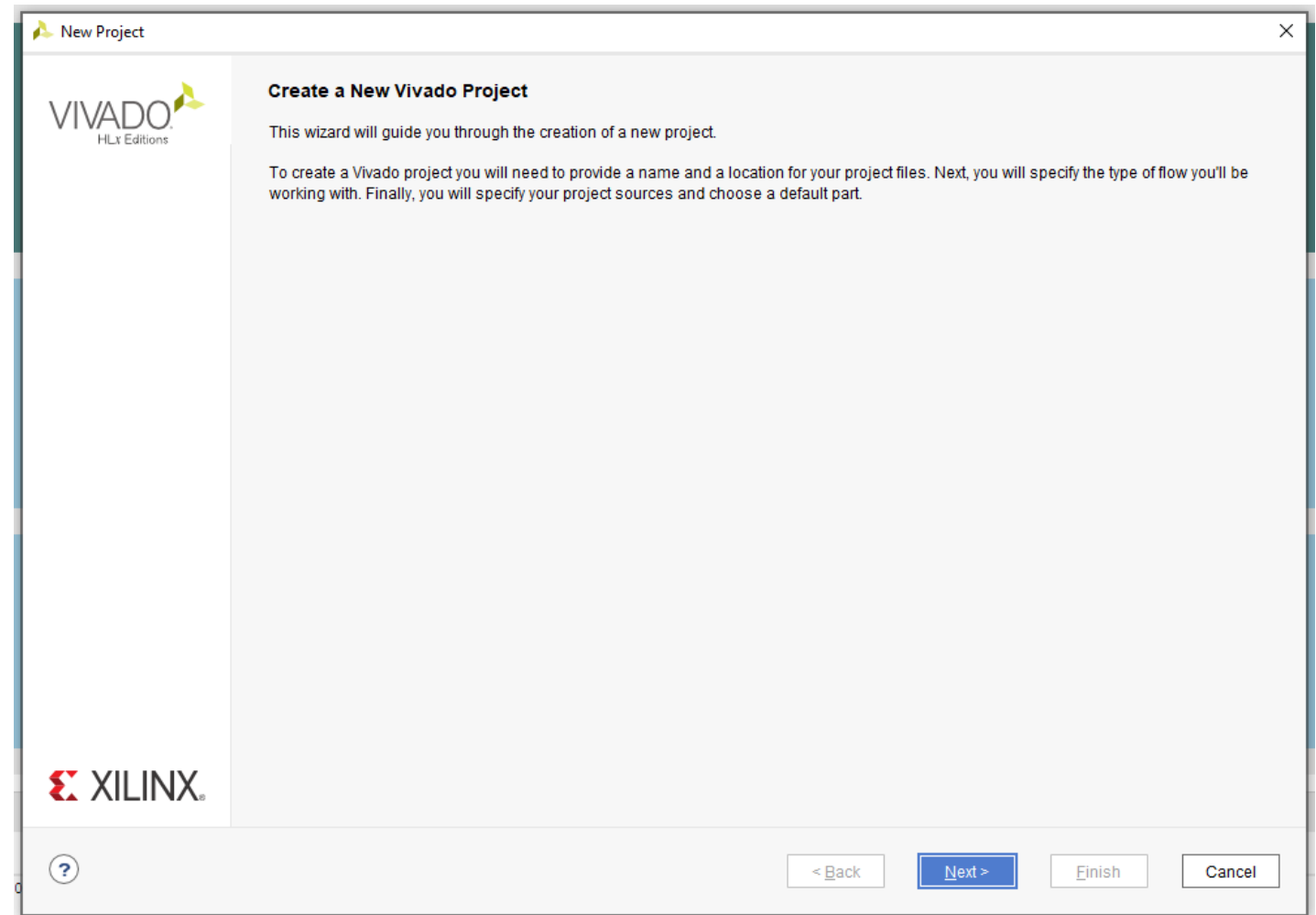
# Lab: FFT Verification

Open Vivado™ and select  
Xhub Stores



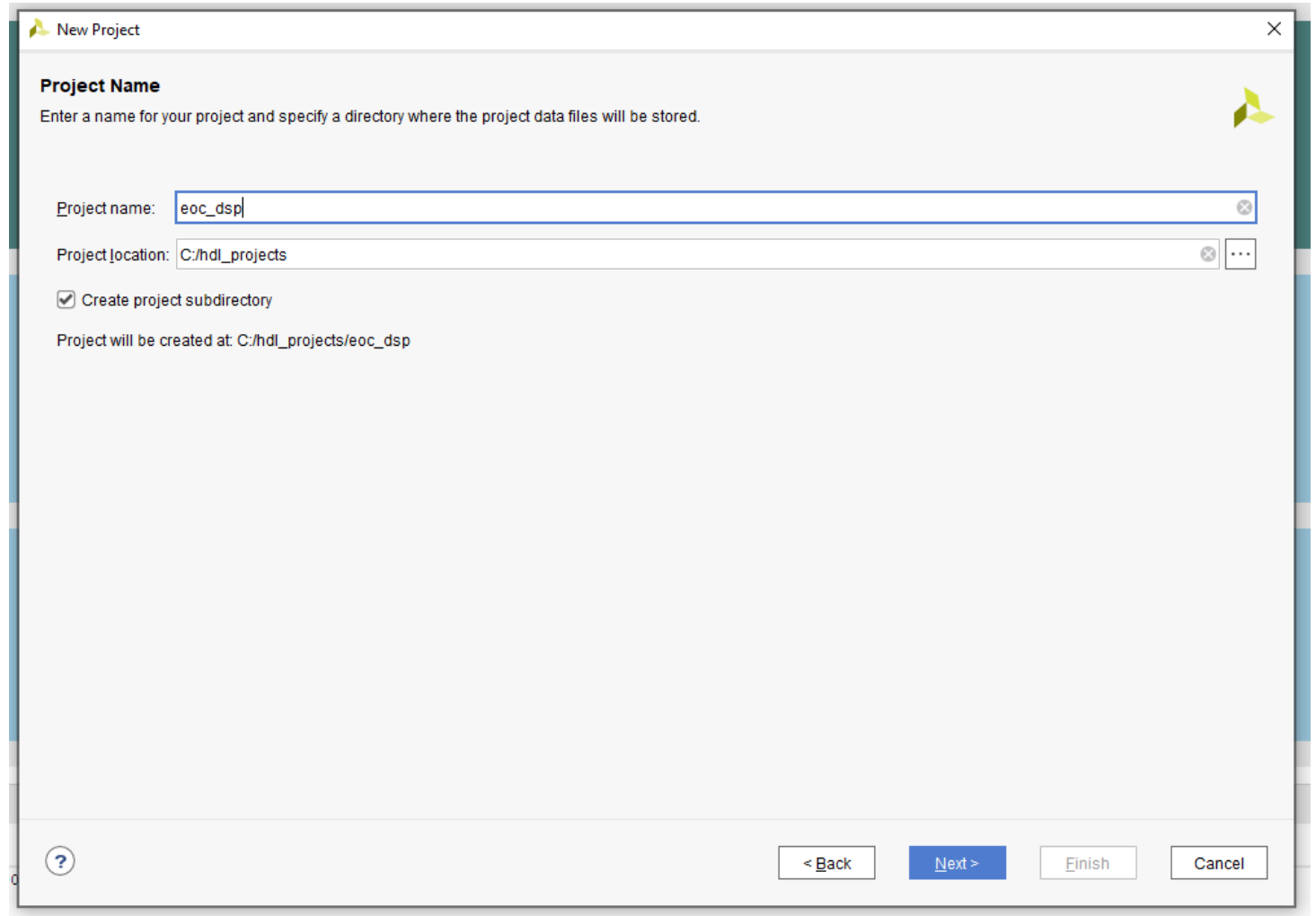
# Lab: FFT Verification

Create a new project



# Lab: FFT Verification

Enter a name and location



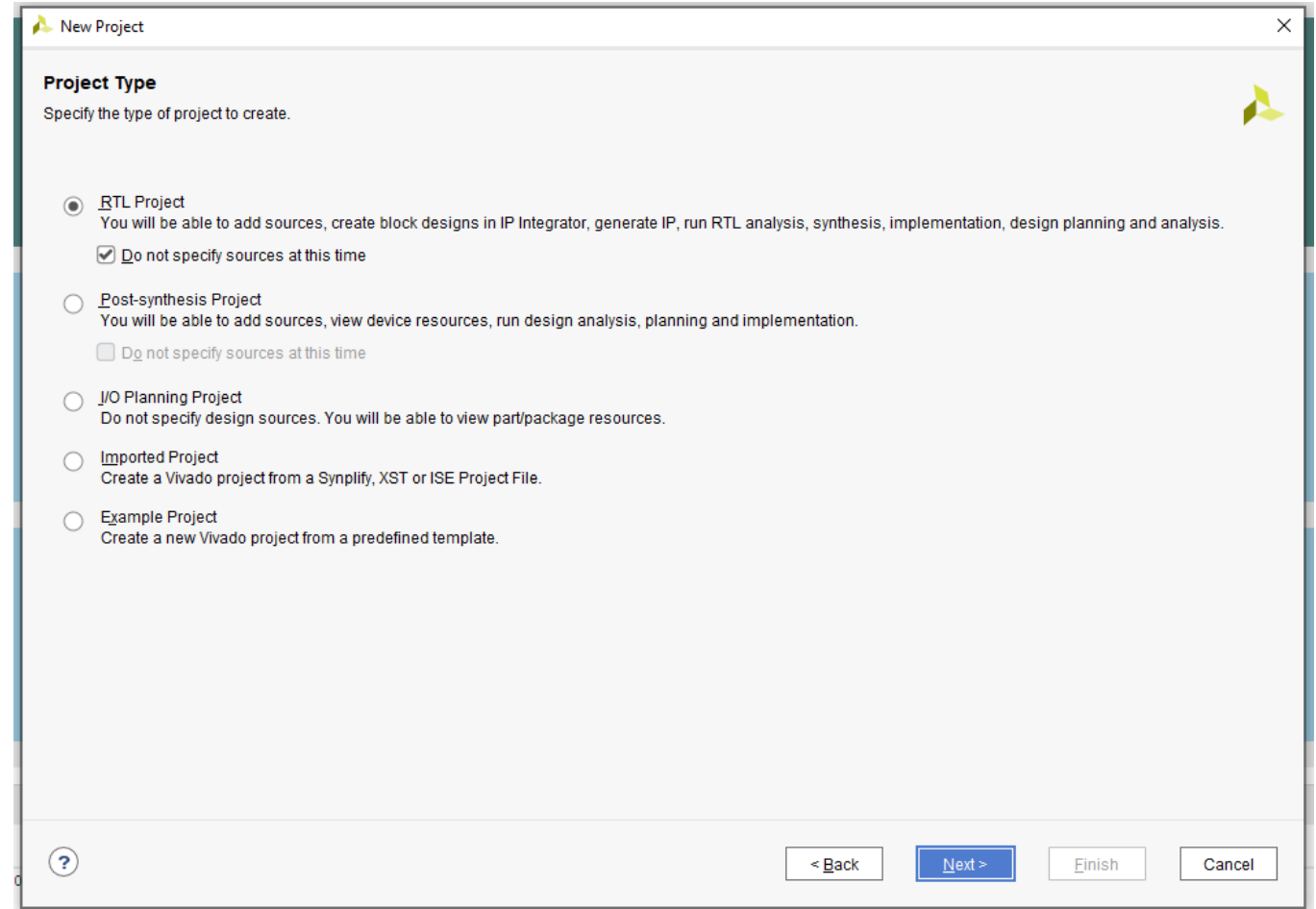
The image shows a 'New Project' dialog box with the following fields and options:

- Project Name**  
Enter a name for your project and specify a directory where the project data files will be stored.
- Project name:** eoc\_dsp
- Project location:** C:/hdl\_projects
- ☒ Create project subdirectory
- Project will be created at: C:/hdl\_projects/eoc\_dsp
- Buttons: < Back, Next >, Finish, Cancel



# Lab: FFT Verification

Select RTL Project



The image shows a 'New Project' dialog box from a software application. The title bar says 'New Project' with a close button. The main area is titled 'Project Type' with the instruction 'Specify the type of project to create.' There are five radio button options: 'RTL Project' (selected), 'Post-synthesis Project', 'I/O Planning Project', 'Imported Project', and 'Example Project'. Each option has a description. Under 'RTL Project', there is a checked checkbox 'Do not specify sources at this time'. At the bottom, there are four buttons: a help button (question mark), '< Back', 'Next >' (highlighted in blue), 'Finish', and 'Cancel'.

**New Project**

**Project Type**  
Specify the type of project to create.

- ☒ **RTL Project**  
You will be able to add sources, create block designs in IP Integrator, generate IP, run RTL analysis, synthesis, implementation, design planning and analysis.  
☒ Do not specify sources at this time
- ☐ **Post-synthesis Project**  
You will be able to add sources, view device resources, run design analysis, planning and implementation.  
☐ Do not specify sources at this time
- ☐ **I/O Planning Project**  
Do not specify design sources. You will be able to view part/package resources.
- ☐ **Imported Project**  
Create a Vivado project from a Synplify, XST or ISE Project File.
- ☐ **Example Project**  
Create a new Vivado project from a predefined template.

? < Back Next > Finish Cancel

# Lab: FFT Verification

Select the ZU Board

New Project ✕






**Default Part**  
Choose a default Xilinx part or board for your project.

Parts | **Boards**

[Reset All Filters](#)

Vendor:  Name:  Board Rev:

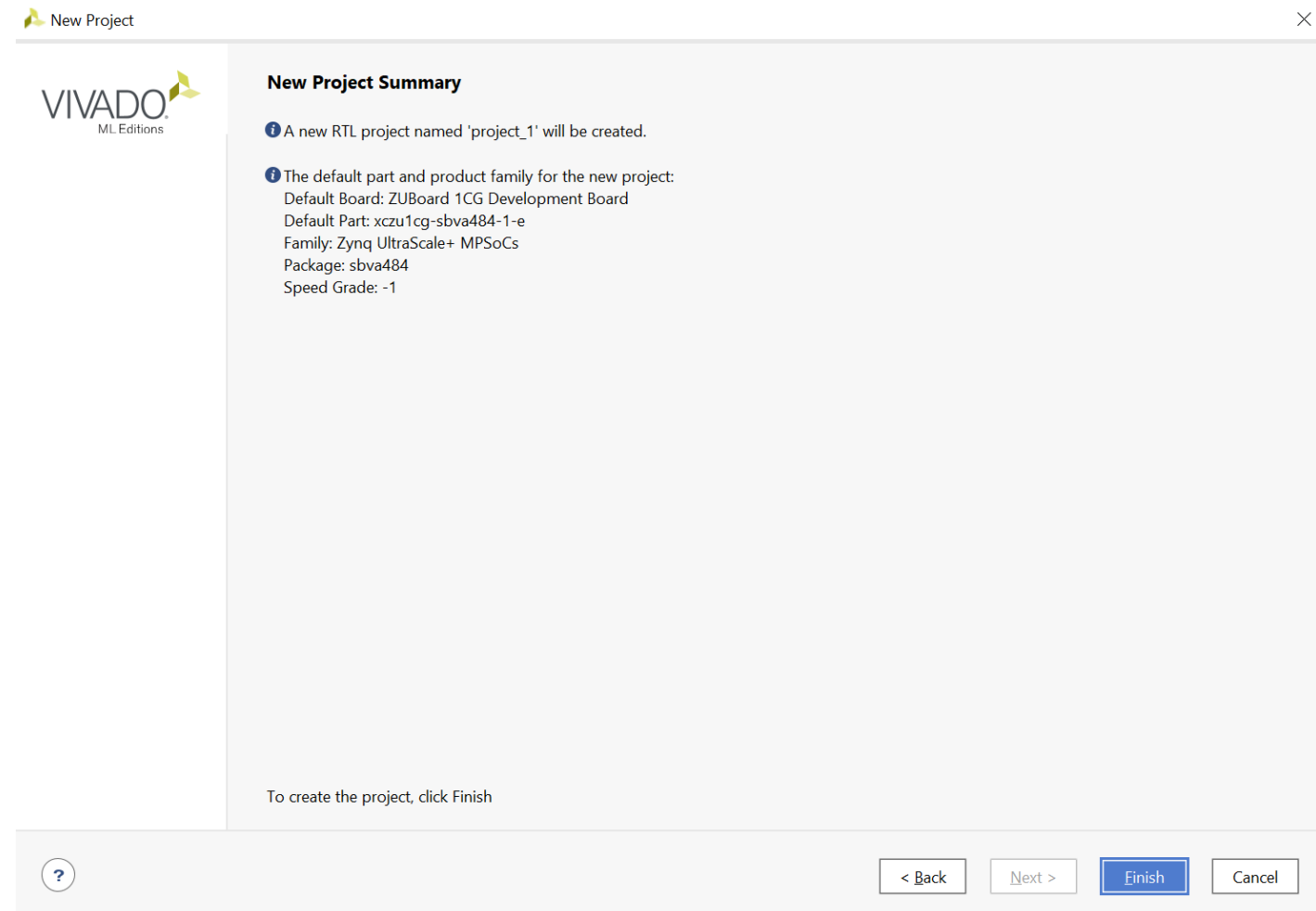
Search:

Display Name	Preview	Status	Vendor	File Version	Part	I/O Pin Count	Board Rev	Availa
<a href="#">Avnet UltraZed-3EG IO Carrier Card</a>		Installed	avnet.com	1.2	xczu3eg-sfva625-1-i	625	1.0	180
<a href="#">Avnet UltraZed-3EG PCIe Carrier Card</a>		Installed	avnet.com	1.3	xczu3eg-sfva625-1-i	625	1.0	180
<a href="#">Avnet UltraZed-7EV SOM</a>		Installed	avnet.com	1.4	xczu7ev-fbvb900-1-i	900	1.0	204
<a href="#">ZedBoard Zynq Evaluation and Development Kit</a> <a href="#">Add Companion Card</a> <a href="#">Connections</a>		Installed	avnet.com	1.4	xc7z020clg484-1	484	d	200
<a href="#">ZUBoard 1CG Development Board</a> <a href="#">Add Companion Card</a> <a href="#">Connections</a>		Installed	avnet.com	1.0	xczu1cg-sbva484-1-e	484	Rev 1	82

?

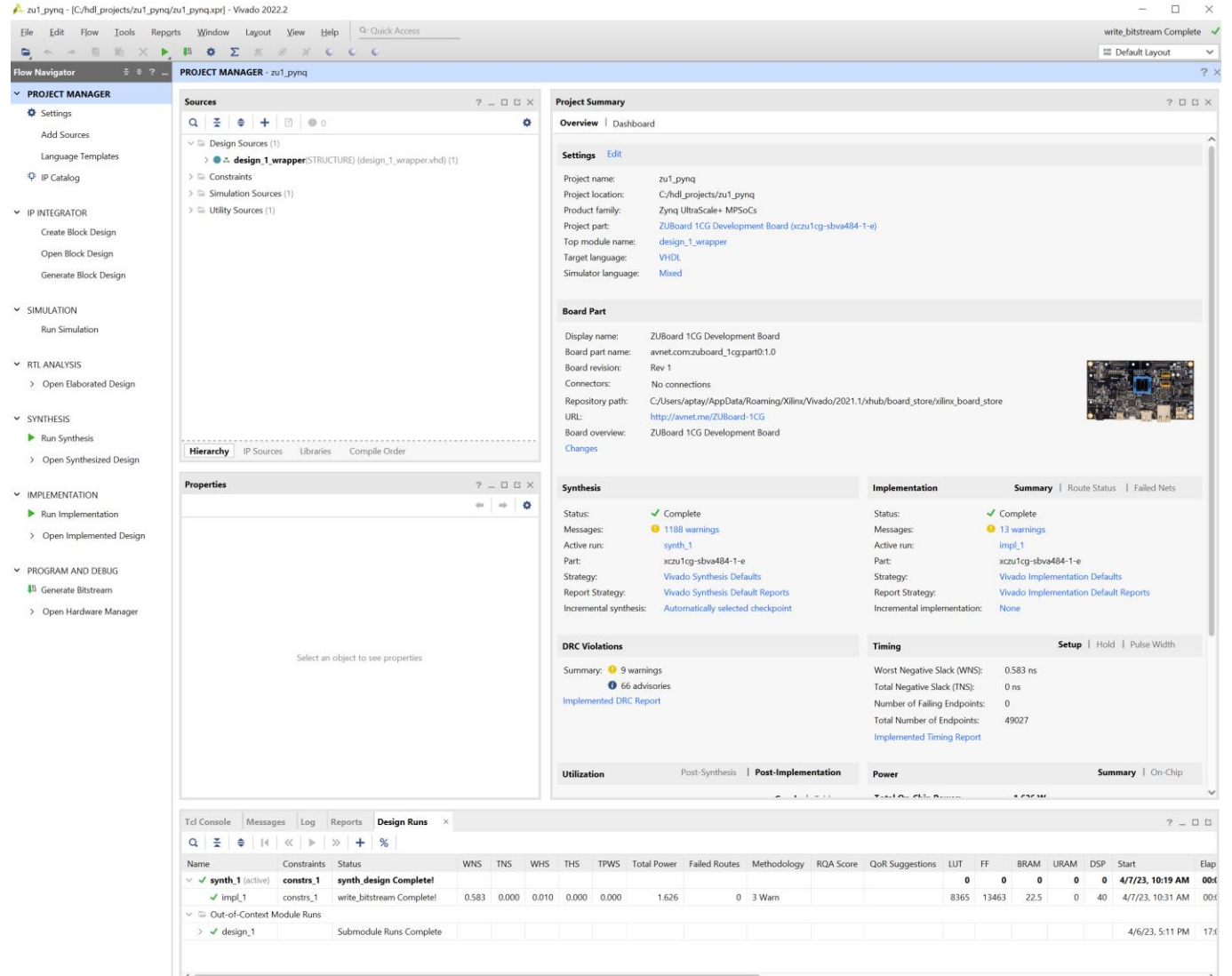
# Lab: FFT Verification

Click Finish to create the project



# Lab: FFT Verification

From the Project  
Manager, select create  
block diagram



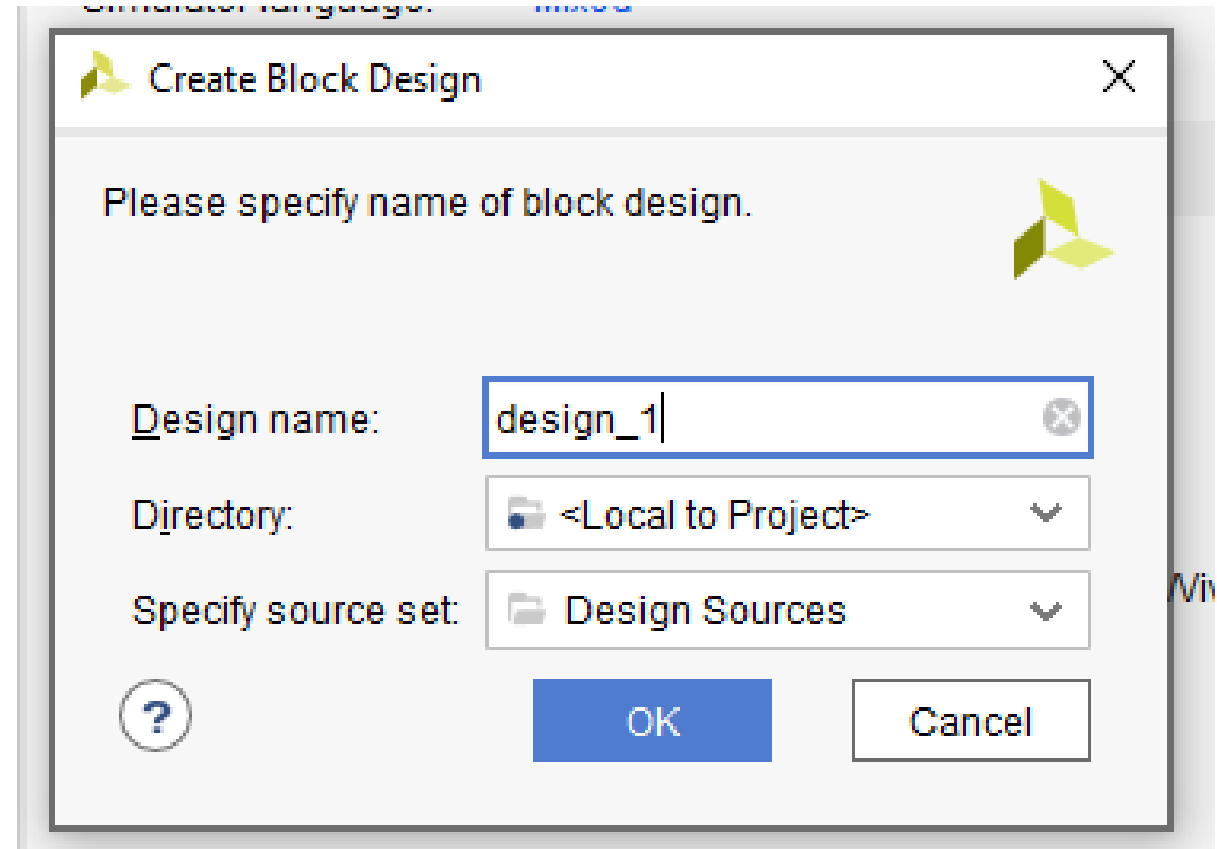
The screenshot displays the Vivado 2022.2 Project Manager interface for a project named 'zu1\_pynq'. The interface is divided into several panes:

- Flow Navigator:** Shows the project workflow steps: Settings, Add Sources, Language Templates, IP Catalog, IP INTEGRATOR, SIMULATION, RTL ANALYSIS, SYNTHESIS, IMPLEMENTATION, and PROGRAM AND DEBUG.
- Sources:** Lists the project sources, including 'design\_1\_wrapper' (STRUCTURE) and 'constraints'.
- Properties:** A pane for viewing the properties of the selected object.
- Project Summary:** Provides an overview of the project, including:
  - Settings:** Project name (zu1\_pynq), Project location (C:/hdl\_projects/zu1\_pynq), Product family (Zynq UltraScale+ MPSoCs), Project part (ZUBoard 1CG Development Board), Top module name (design\_1\_wrapper), Target language (VHDL), and Simulator language (Mixed).
  - Board Part:** Display name (ZUBoard 1CG Development Board), Board part name (avnet.com:zuboard\_1cg:part0:1.0), Board revision (Rev 1), Connectors (No connections), Repository path (C:/Users/aplay/AppData/Roaming/Xilinx/Vivado/2021.1/hub/board\_store/xilinx\_board\_store), URL (http://avnet.me/ZUBoard-1CG), and Board overview (ZUBoard 1CG Development Board).
  - Synthesis:** Status (Complete), Messages (1188 warnings), Active run (synth\_1), Part (xczu1cg-sbva484-1-e), Strategy (Vivado Synthesis Defaults), Report Strategy (Vivado Synthesis Default Reports), and Incremental synthesis (Automatically selected checkpoint).
  - Implementation:** Status (Complete), Messages (13 warnings), Active run (impl\_1), Part (xczu1cg-sbva484-1-e), Strategy (Vivado Implementation Defaults), Report Strategy (Vivado Implementation Default Reports), and Incremental implementation (None).
  - DRC Violations:** Summary (9 warnings, 66 advisories), Implemented DRC Report.
  - Timing:** Setup, Hold, Pulse Width. Worst Negative Slack (WNS): 0.583 ns, Total Negative Slack (TNS): 0 ns, Number of Failing Endpoints: 0, Total Number of Endpoints: 49027.
  - Utilization:** Post-Synthesis, Post-Implementation, Power, Summary, On-Chip.
- Design Runs:** A table showing the status of various design runs.

Name	Constraints	Status	WNS	TNS	WHS	THS	TPWS	Total Power	Failed Routes	Methodology	RQA Score	QoR Suggestions	LUT	FF	BRAM	URAM	DSP	Start	Elap
✓ synth_1 (active)	constrs_1	synth_design Complete!											0	0	0	0	0	4/7/23, 10:19 AM	00:00
✓ impl_1	constrs_1	write_bitstream Complete!	0.583	0.000	0.010	0.000	0.000	1.626	0	3 Warn			8365	13463	22.5	0	40	4/7/23, 10:31 AM	00:00
Out-of-Context Module Runs																			
✓ design_1		Submodule Runs Complete																4/6/23, 5:11 PM	17:00

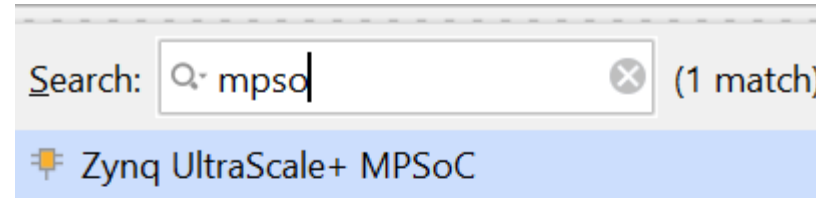
# Lab: FFT Verification

Leave, defaults unchanged and click OK



# Lab: FFT Verification

Click on + and in the  
search bar type in mpsoc  
and press enter



ENTER to select, ESC to cancel, Ctrl+Q for IP details

# Lab: FFT Verification

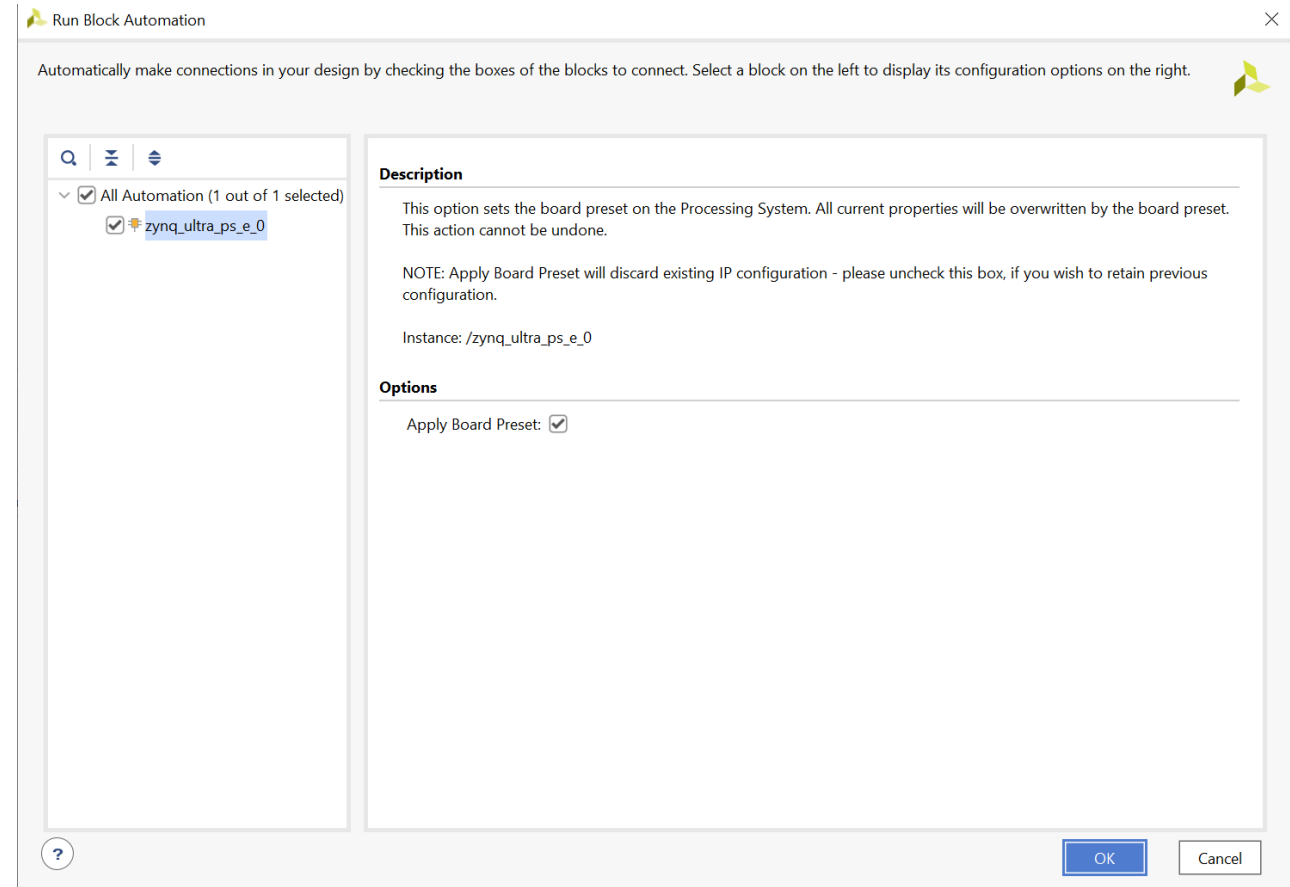
✦ Designer Assistance available. [Run Block Automation](#)

Run the block automation



# Lab: FFT Verification

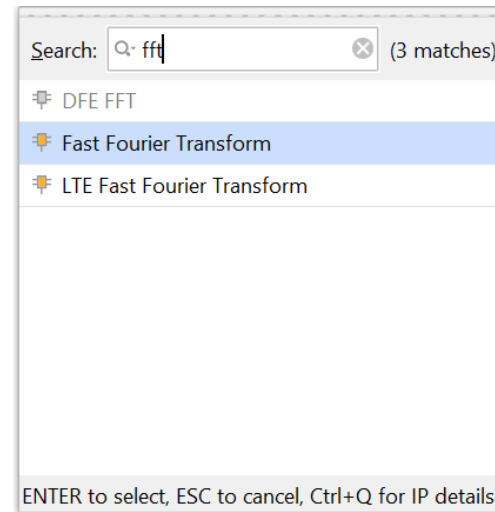
Leave the settings as default and  
click OK





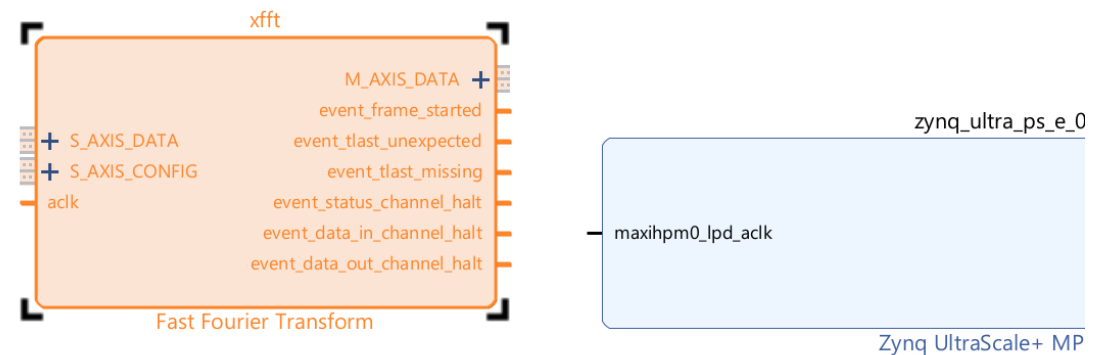
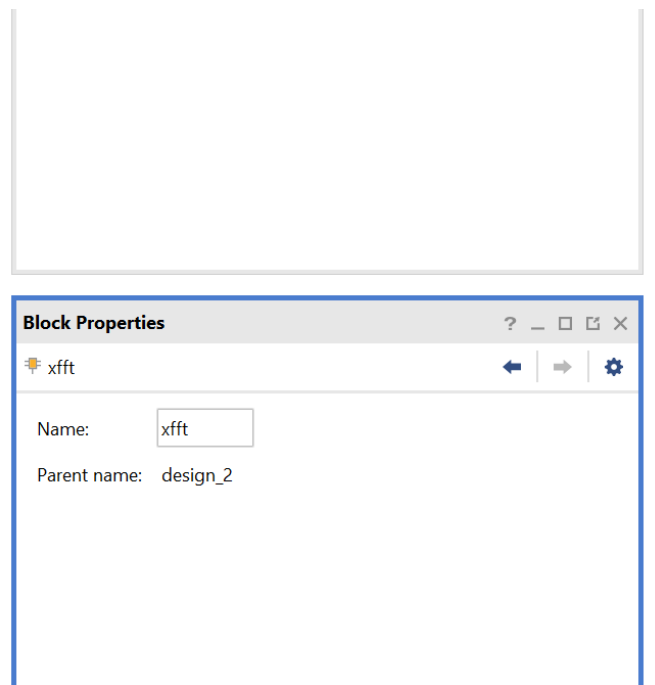
# Lab: FFT Verification

Click on + and add in the  
FFT



# Lab: FFT Verification

Click on the Fast Fourier Transform and change its name to xfft. Double click on the block to customize it.

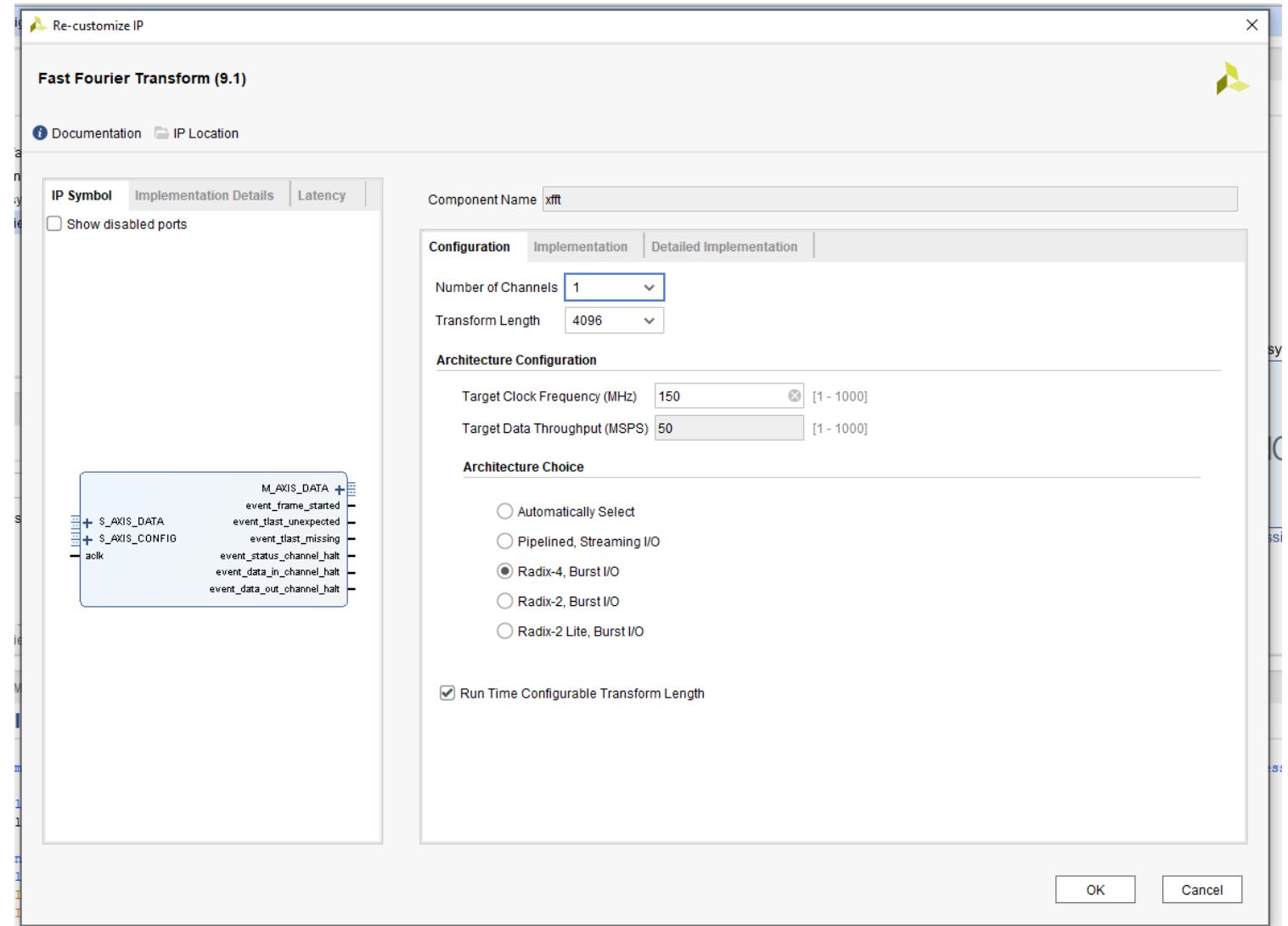


# Lab: FFT Verification

On the configuration tab, select

- Transform length 4096
- Radix-4 Burst I/O
- Target Frequency 150Mhz
- Enable Run Time

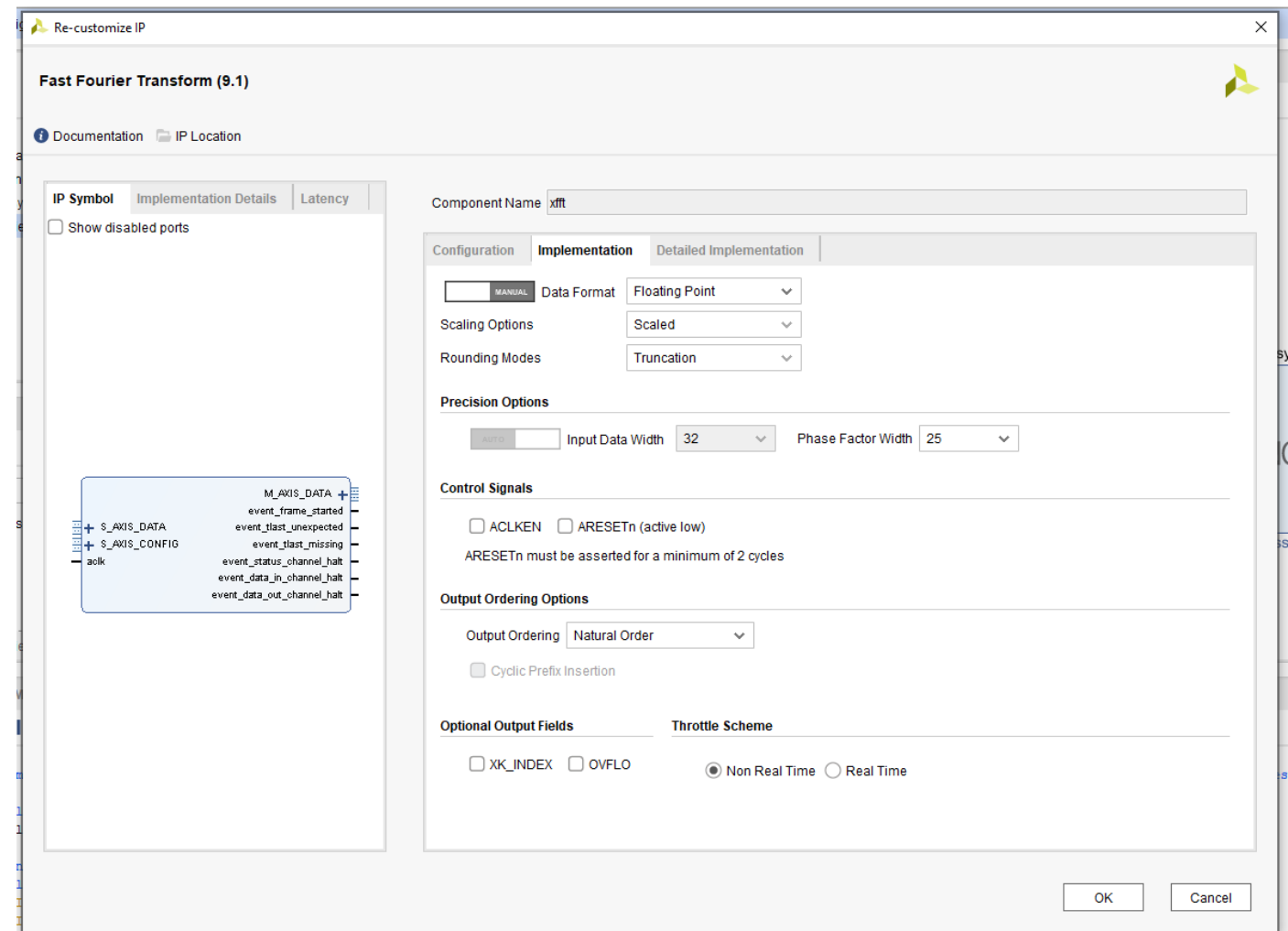
Configurable transform length



# Lab: FFT Verification

On the implementation tab select

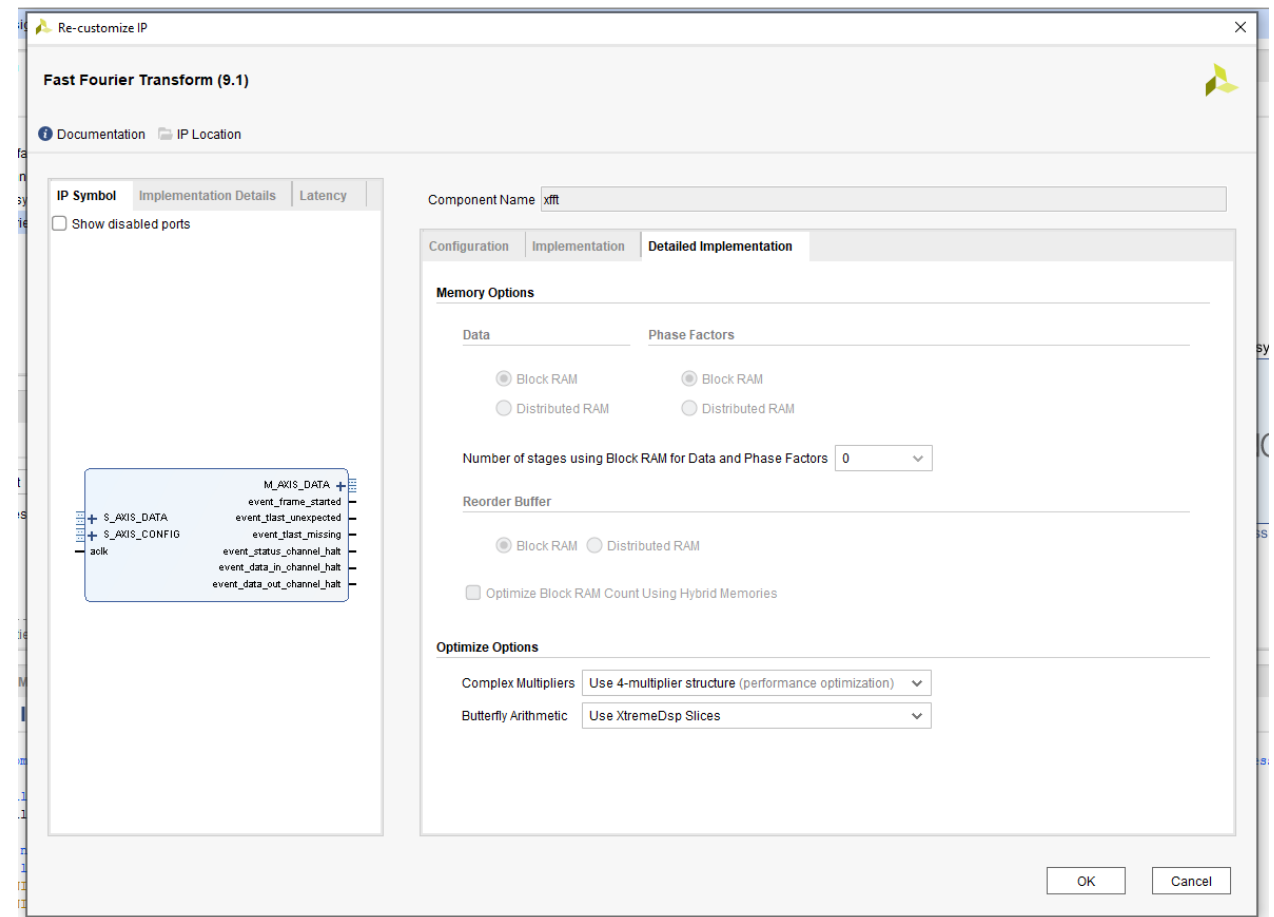
- Floating Point
- Phase Factor Width 25
- Output Ordering Natural
- Non-Real Time Throttle scheme



# Lab: FFT Verification

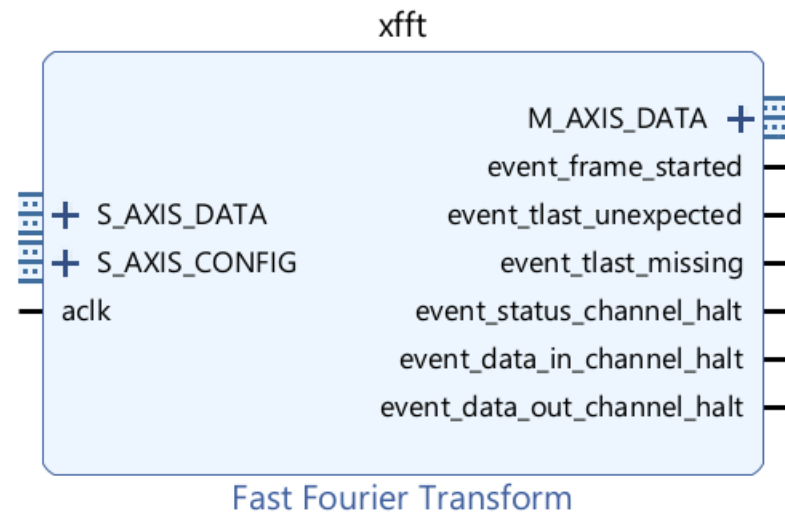
On the Detailed Implementation tab  
select

- Use 4-Multiplier Structure
- Use XtremeDSP Slices



# Lab: FFT Verification

Double click on the Processing System to reconfigure it



# Lab: FFT Verification

On the clocking tab change the frequency of clock one to 250MHz

Re-customize IP

**Zynq UltraScale+ MPSoC (3.4)**

Documentation Presets IP Location

Page Navigator

- ☐ Switch To Advanced Mo
- PS UltraScale+ Block Design
- I/O Configuration
- Clock Configuration**
- DDR Configuration
- PS-PL Configuration

**Clock Configuration**

Input Clocks Output Clocks

☒ Enable Manual Mode

> PLL Options

Search:

Name	Source	FracEn	Requested Freq (MHz)	Divisor 0	Divisor 1	Actual Frequency (MHz)	Range
Low Power Domain Clocks							
Processor/Memory Clocks							
CPU_R5	IO		500	3		500.000000	0.0000...
Peripherals/IO Clocks							
PL Fabric Clocks							
<input checked="" type="checkbox"/> PL0	IO		100	2	3	250.000000	0.0000...
<input type="checkbox"/> PL1	RP		100	30	1	50.000000	0.0000...
<input type="checkbox"/> PL2	RP		100	60	1	25.000000	0.0000...
<input type="checkbox"/> PL3	RP		100	10	1	150.000000	0.0000...
System Debug Clocks							
Full Power Domain Clocks							
Processor/Memory Clocks							
ACPU	AP	<input type="checkbox"/>	1200	1		1200.000000	0.0000...

OK Cancel

# Lab: FFT Verification

On the Interrupts Tab enable the  
IRQ[0:7]

Re-customize IP

**Zynq UltraScale+ MPSoC (3.4)**

Documentation Presets IP Location

**Page Navigator**

- Switch To Advanced Mo
- PS UltraScale+ Block Design
- I/O Configuration
- Clock Configuration
- DDR Configuration
- PS-PL Configuration**

**PS-PL Configuration**

Search: Q-

Name	Select
General	
Interrupts	
PL to PS	
IRQ0[0-7]	1
IRQ1[0-7]	0
APU Legacy Interr...	<input type="checkbox"/>
RPU Legacy Interr...	<input type="checkbox"/>
PS to PL	
Fabric Reset Enable	<input checked="" type="checkbox"/>
Address Fragmentation	
Others	
PS-PL Interfaces	
Master Interface	
Slave Interface	
Debug	

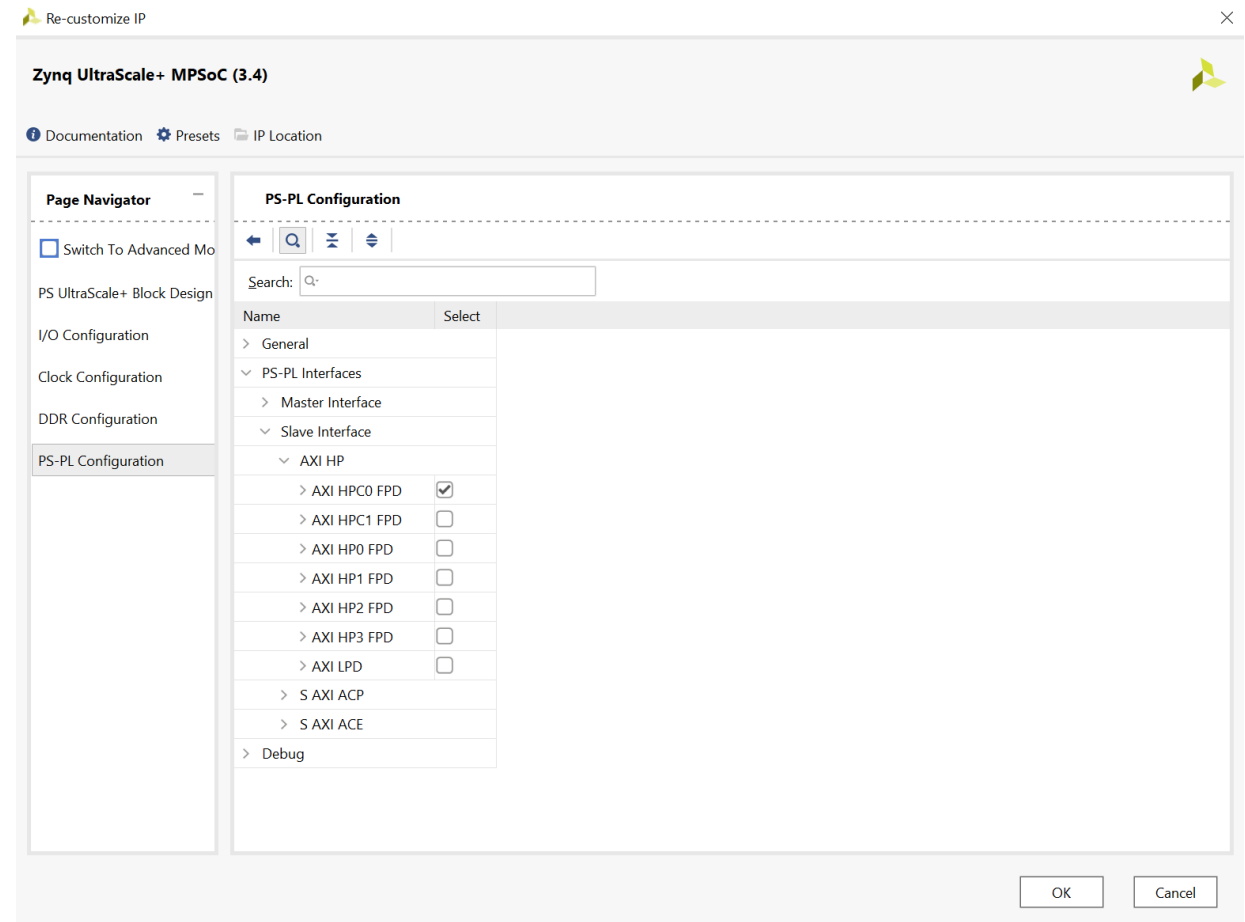
OK Cancel



# Lab: FFT Verification

On the PS/PL interface select the HP  
Slave AXI Interface

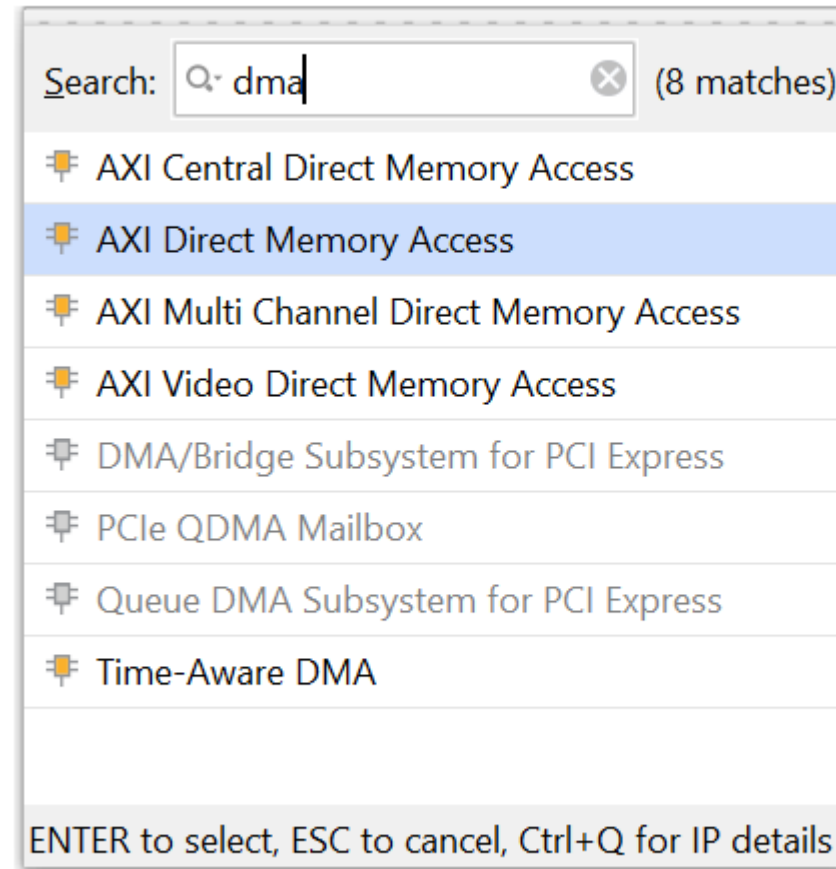
Enable AXI HPC0 FPD Interface



# Lab: FFT Verification

Click + and select AXI

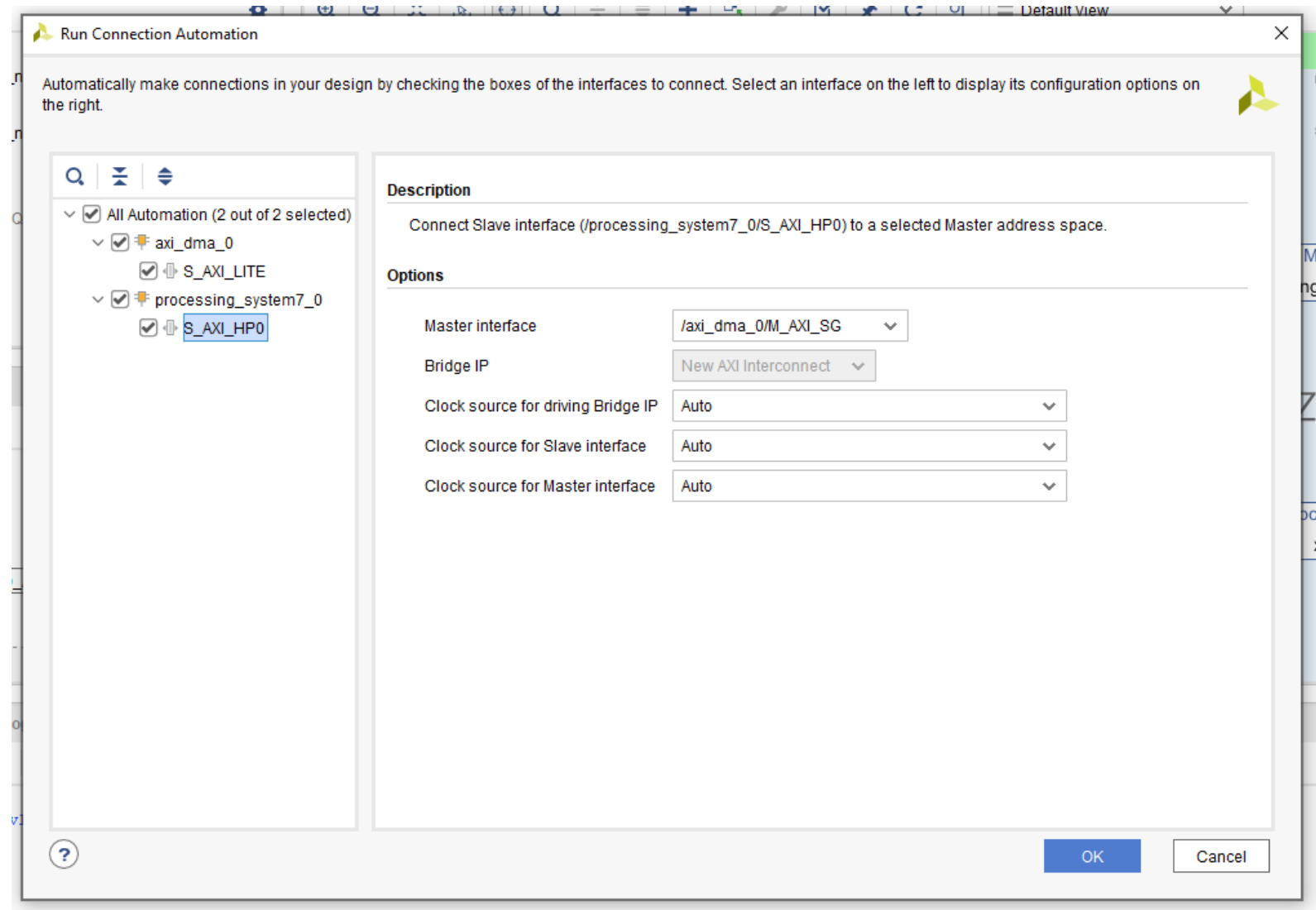
Direct Memory Access



# Lab: FFT Verification

Run the connection automation.

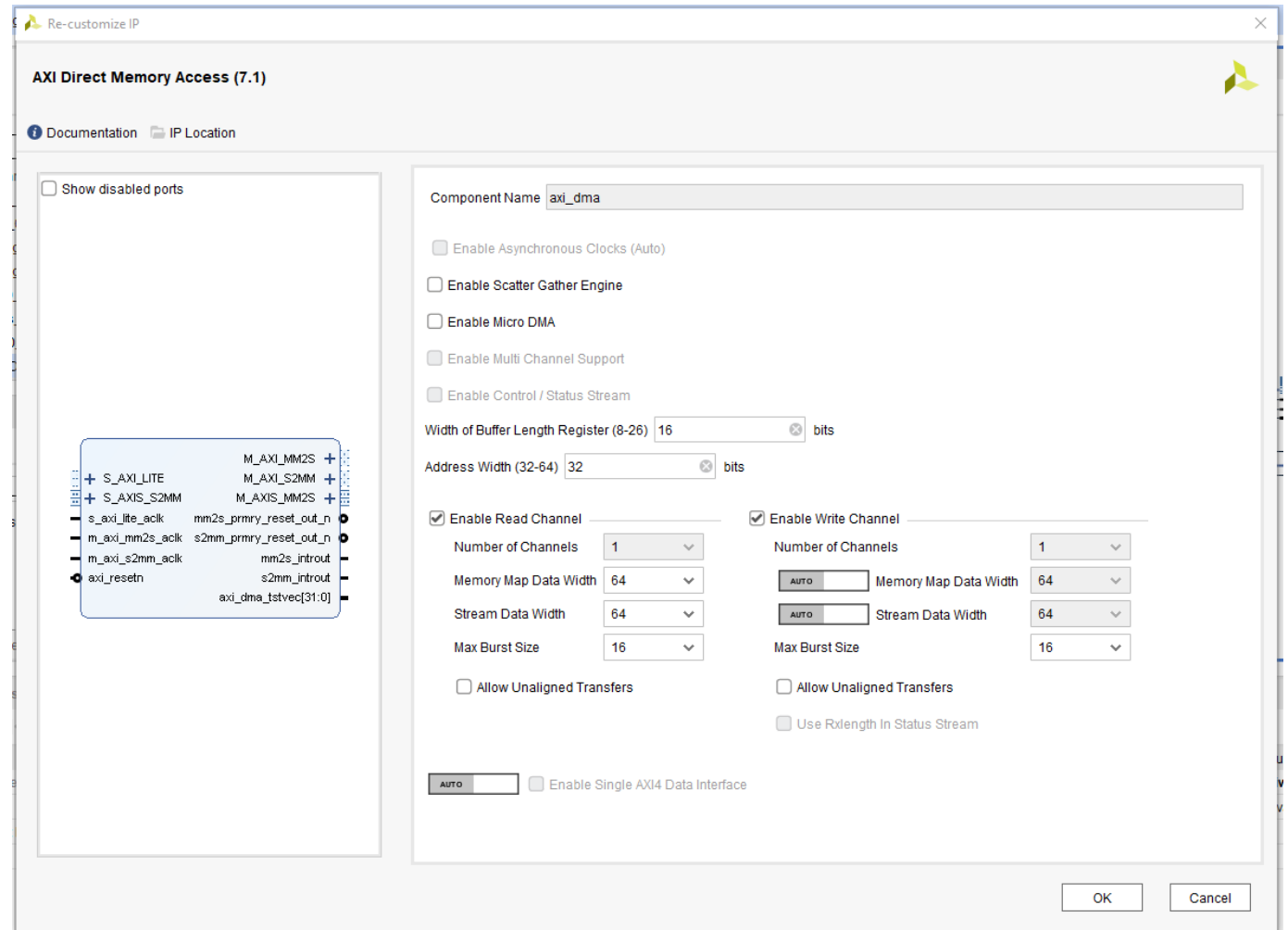
Leave the defaults as standard and click OK.



# Lab: FFT Verification

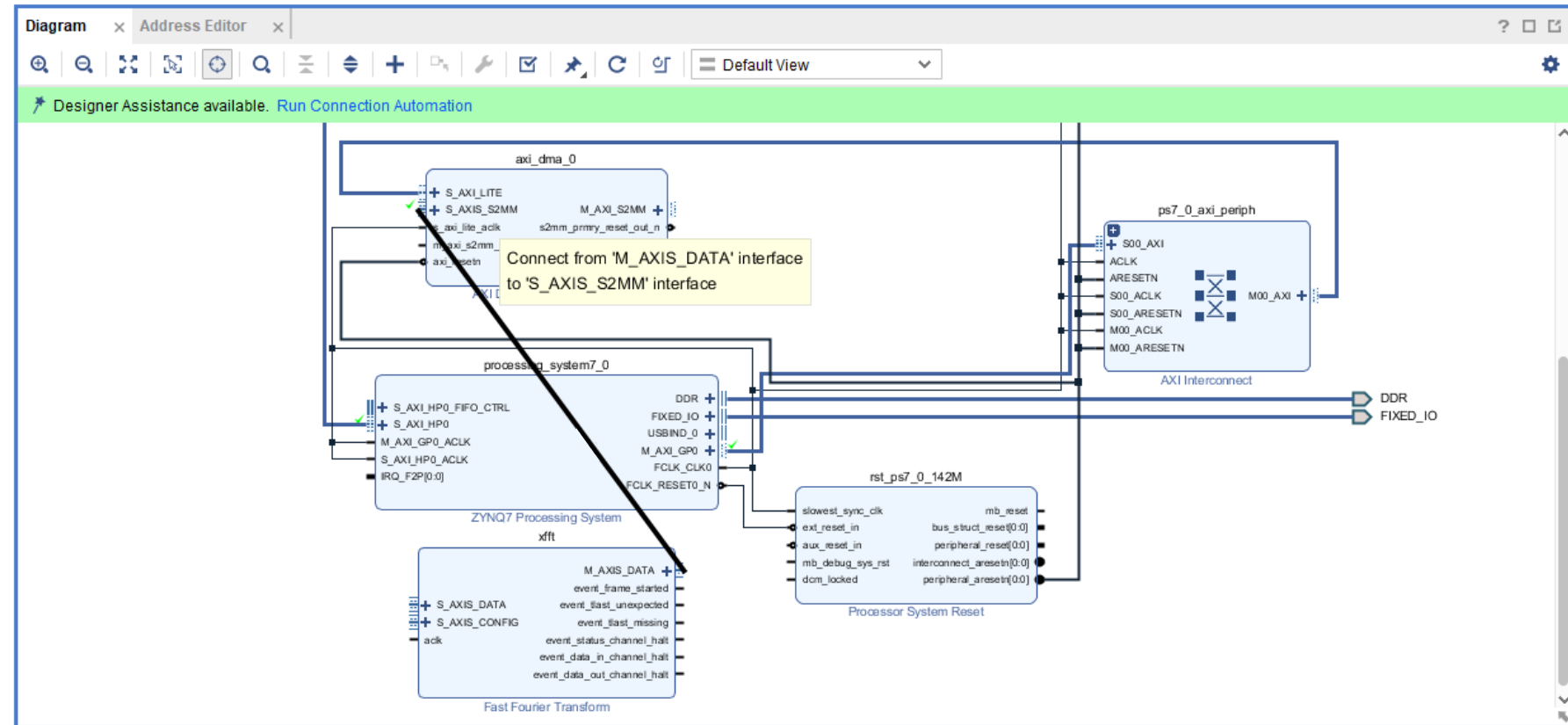
Select the DMA, double click on it and configure it

- Width of Buffer length 16
- Stream data width 64
- Max burst size 16

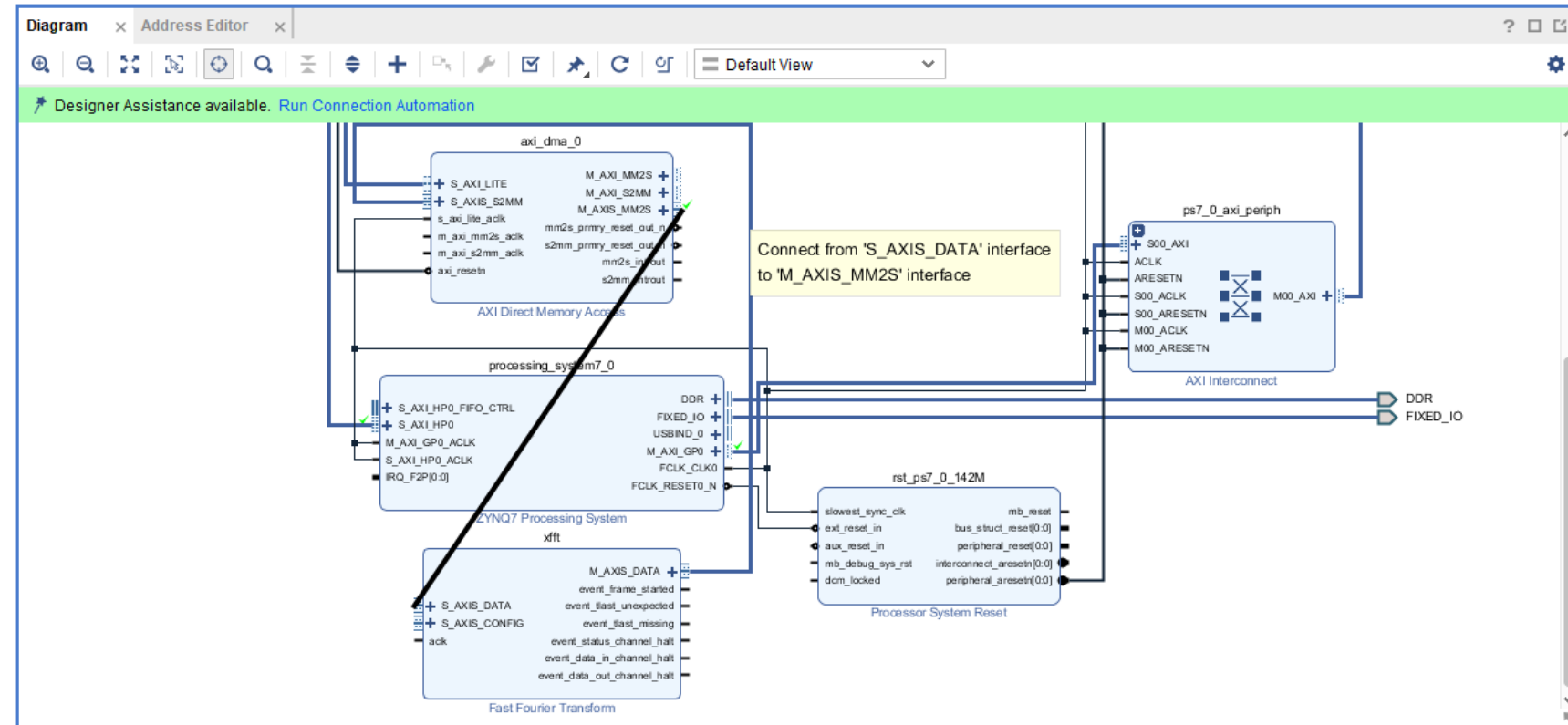


# Lab: FFT Verification

Connect the xFFT M  
 AXIS data to the  
 DMA, S AXIS S2MM  
 port

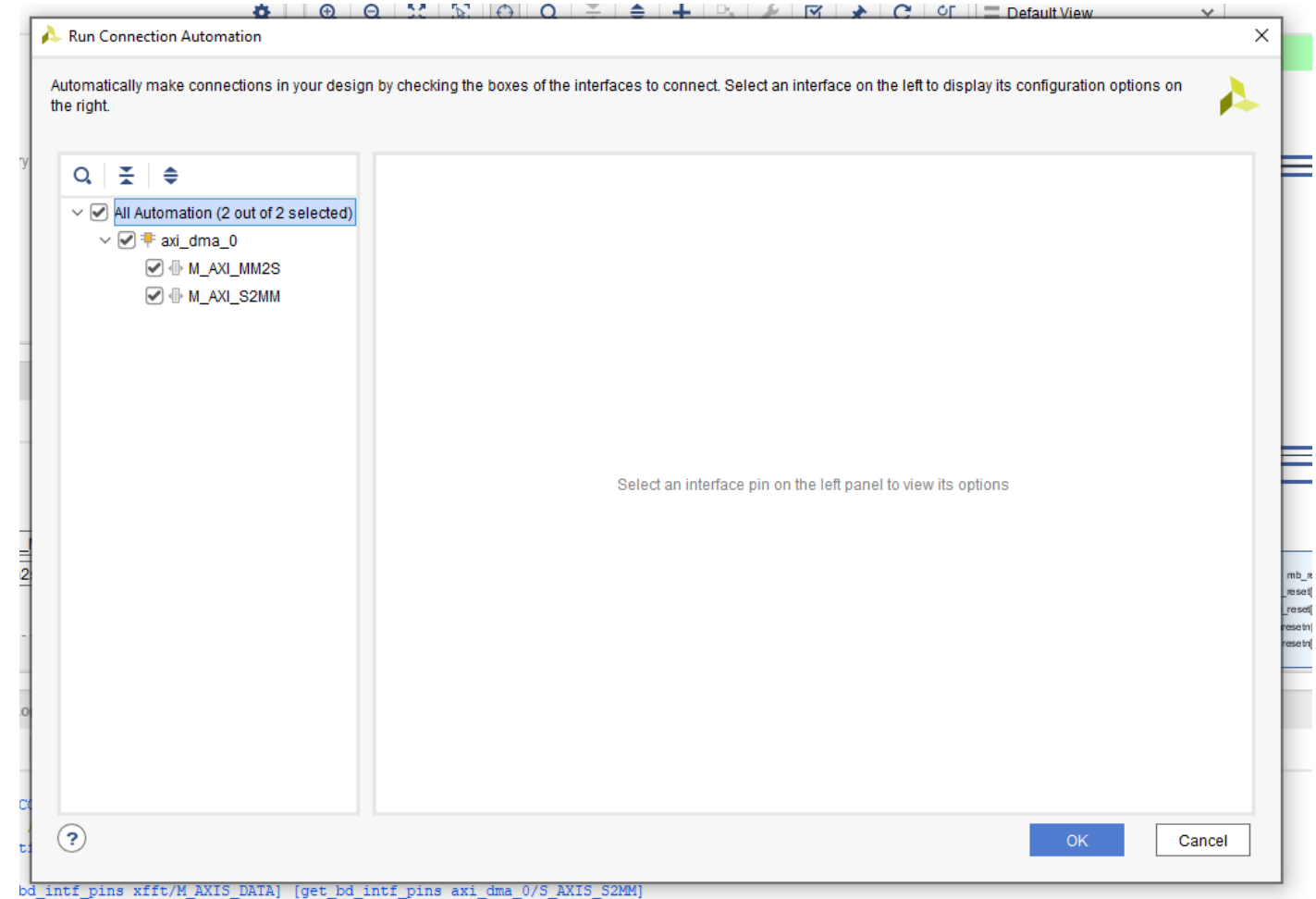


# Connect the DMA M AXIS MM2S to the xFFT S AXIS Data



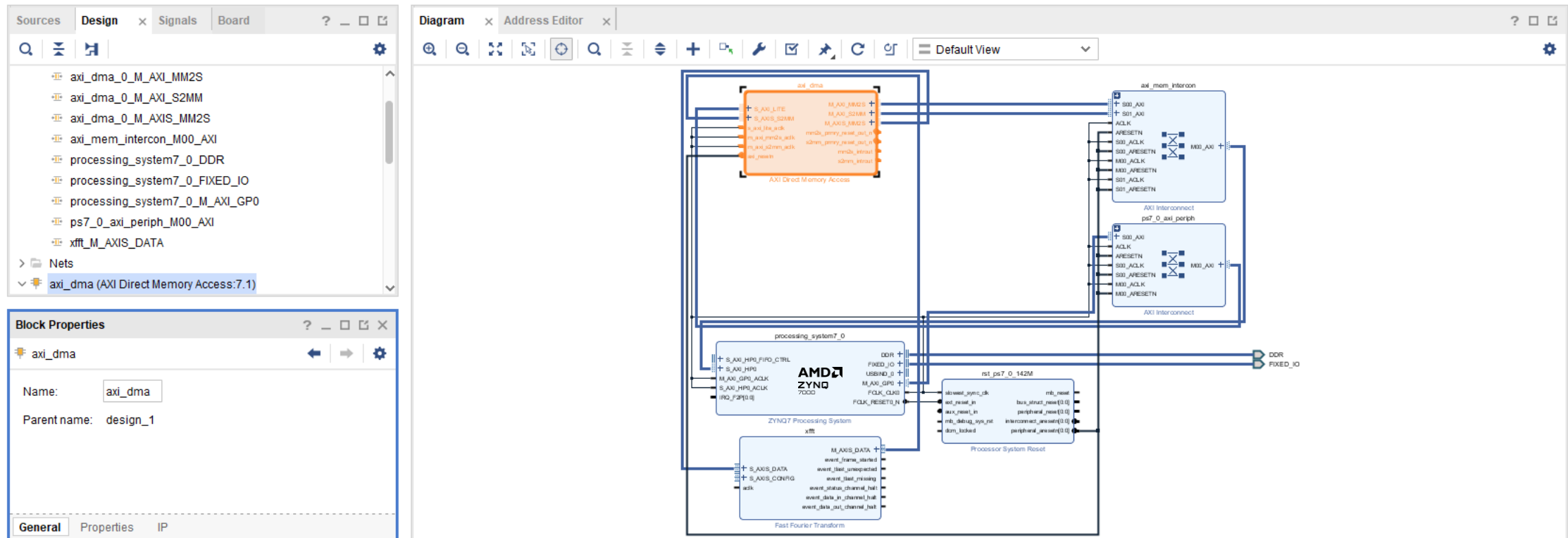
# Lab: FFT Verification

Run the connection automation



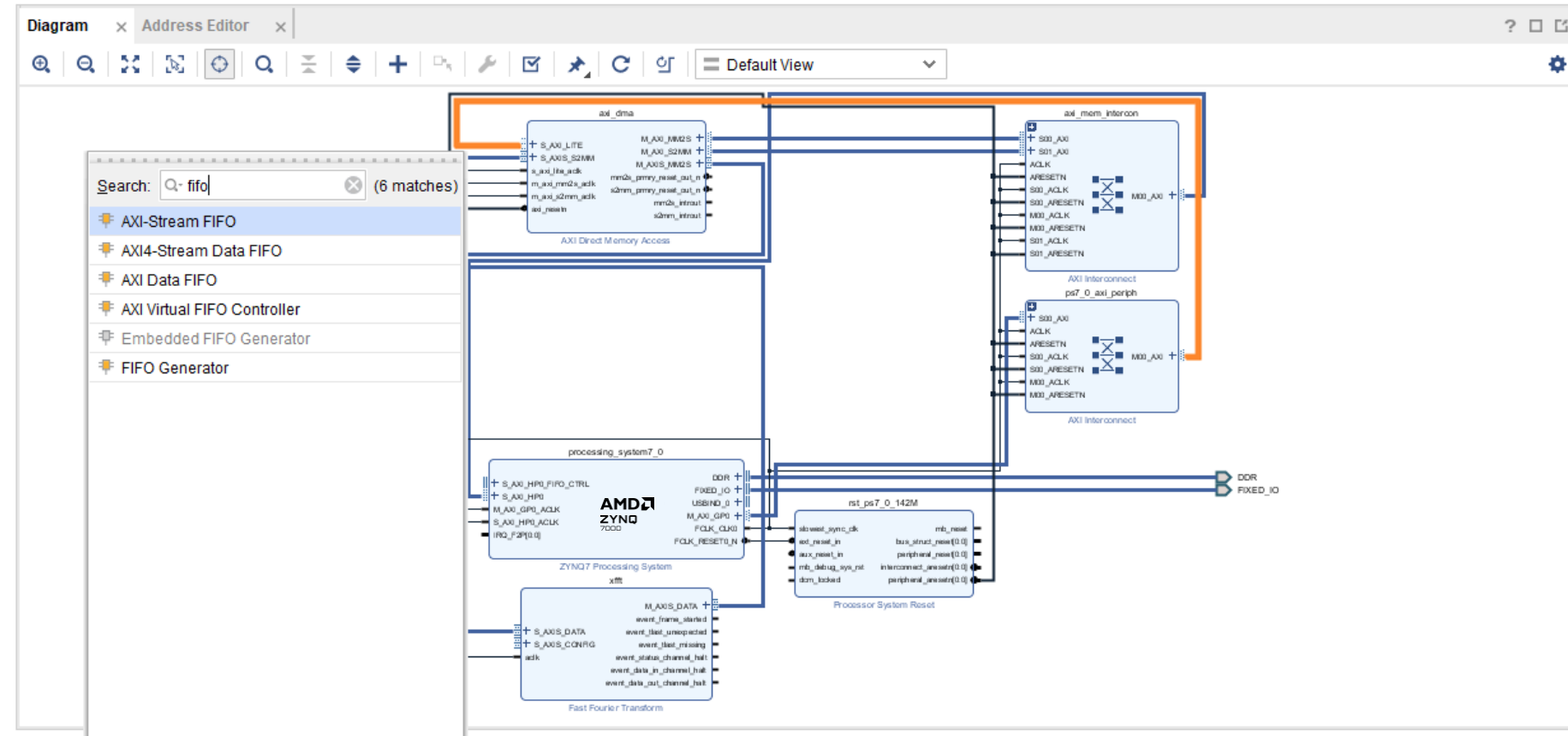
# Lab: FFT Verification

The diagram should look like below





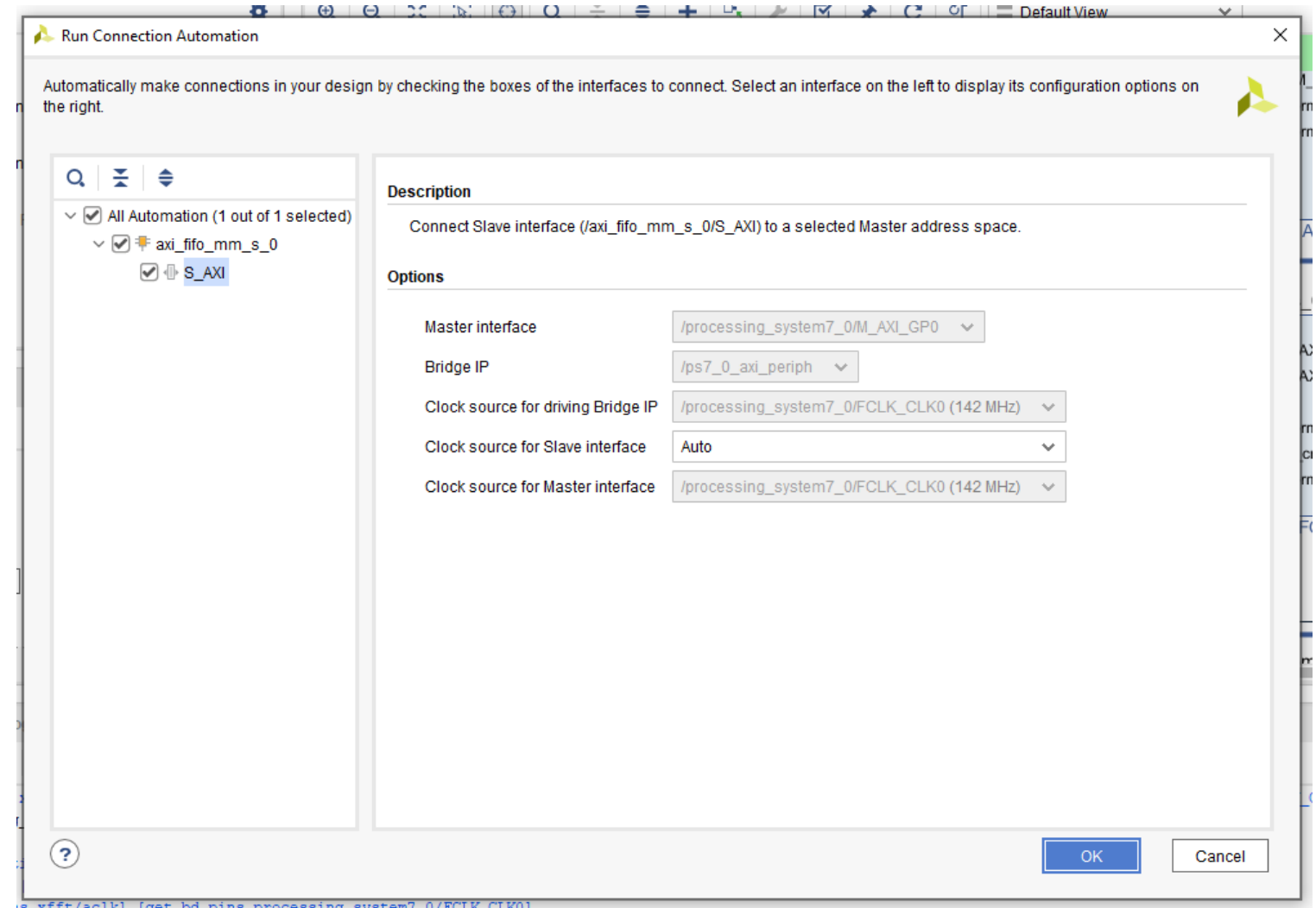
Click on + and add  
in an AXI-Stream  
FIFO



# Lab: FFT Verification

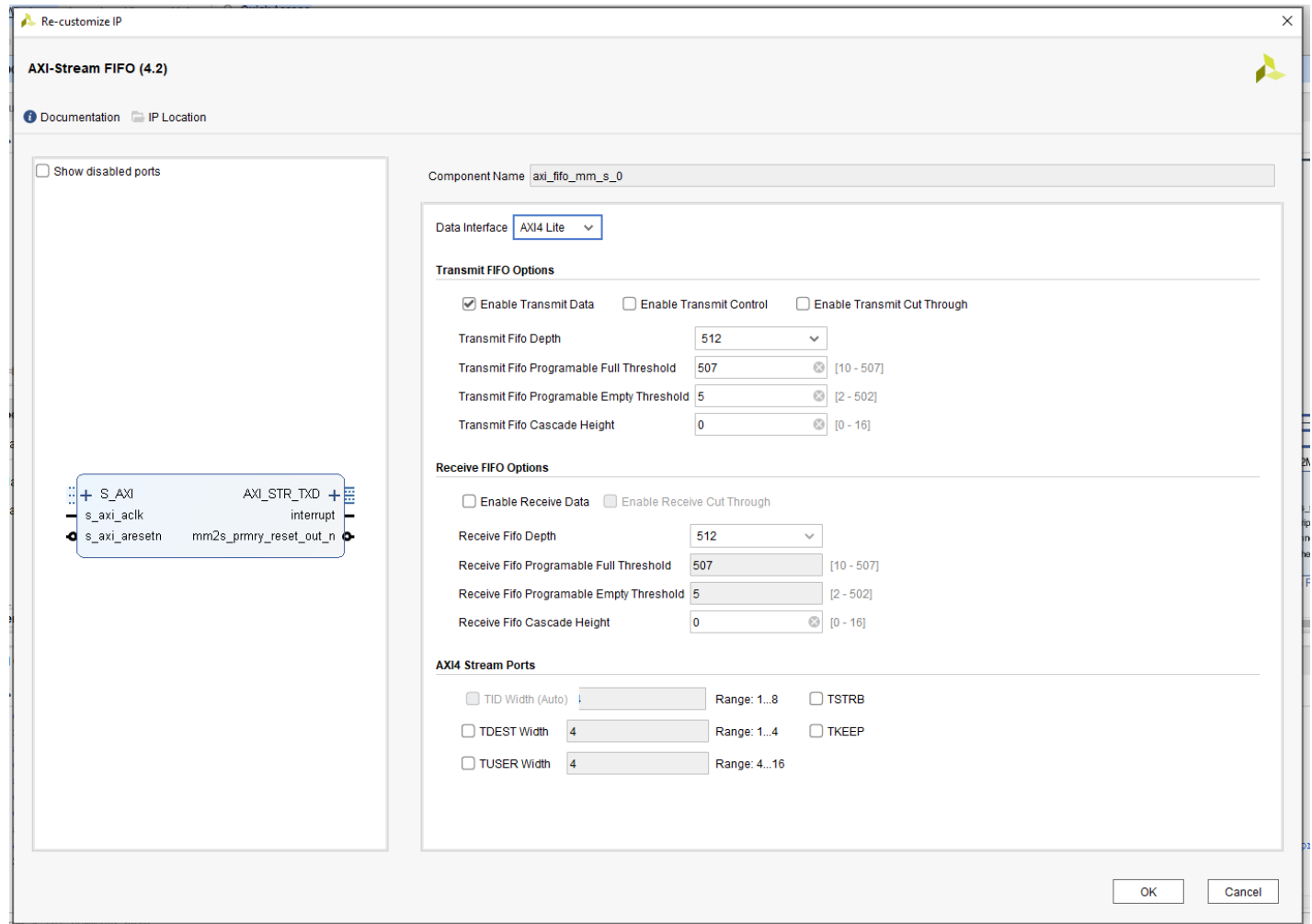
Run the connection

automation and click on OK



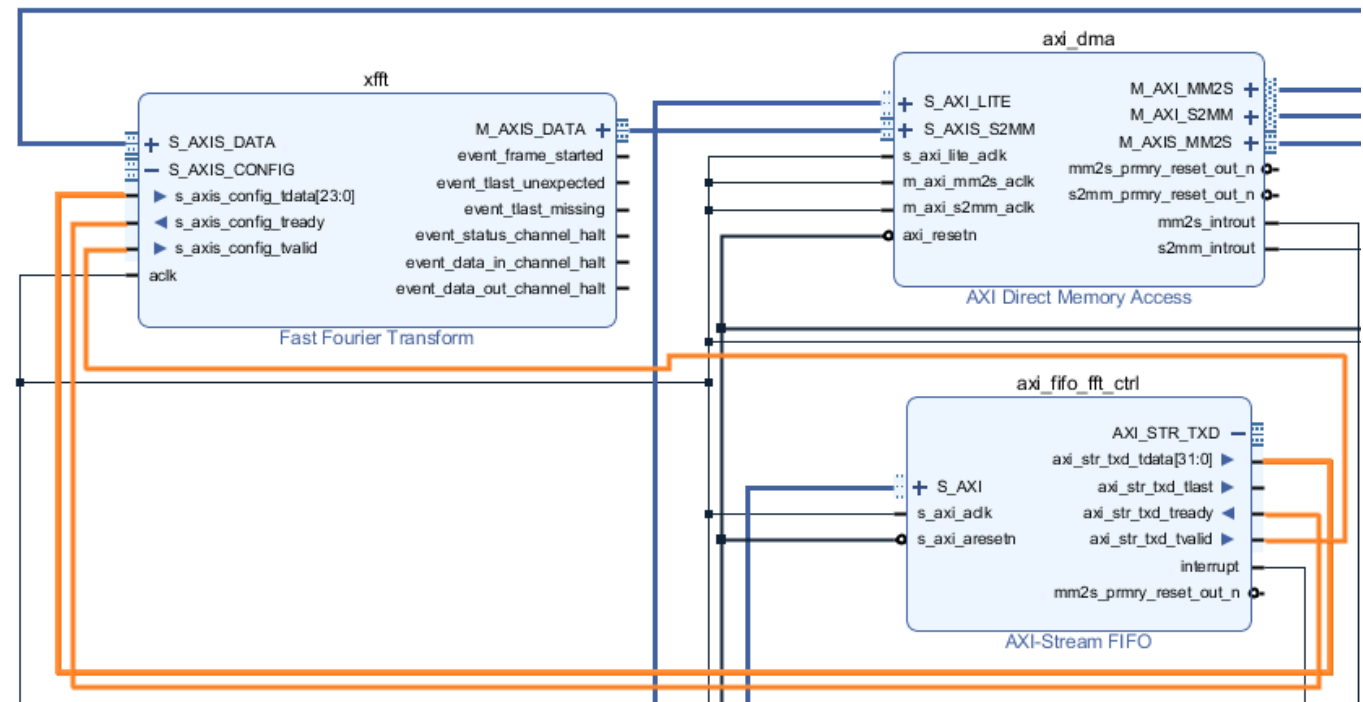
# Lab: FFT Verification

Double click on the AXI Stream FIFO and configure it to have an AXI Lite Interface and only enable the Transmit Data Interface leave all else unchanged.



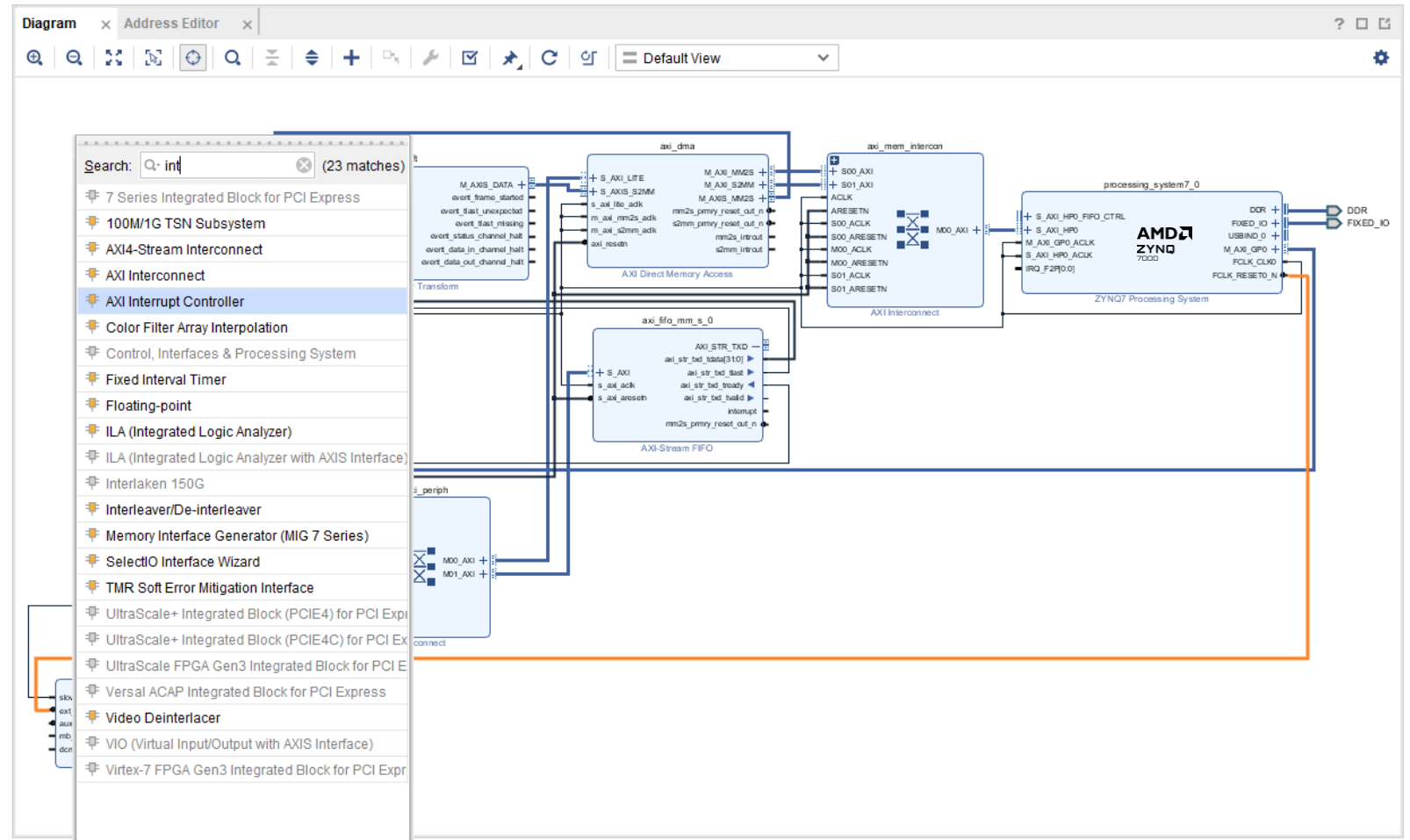
# Lab: FFT Verification

Connect the AXI STR TXD tdata, tlast and tvalid signals to the xFFT S AXIS config



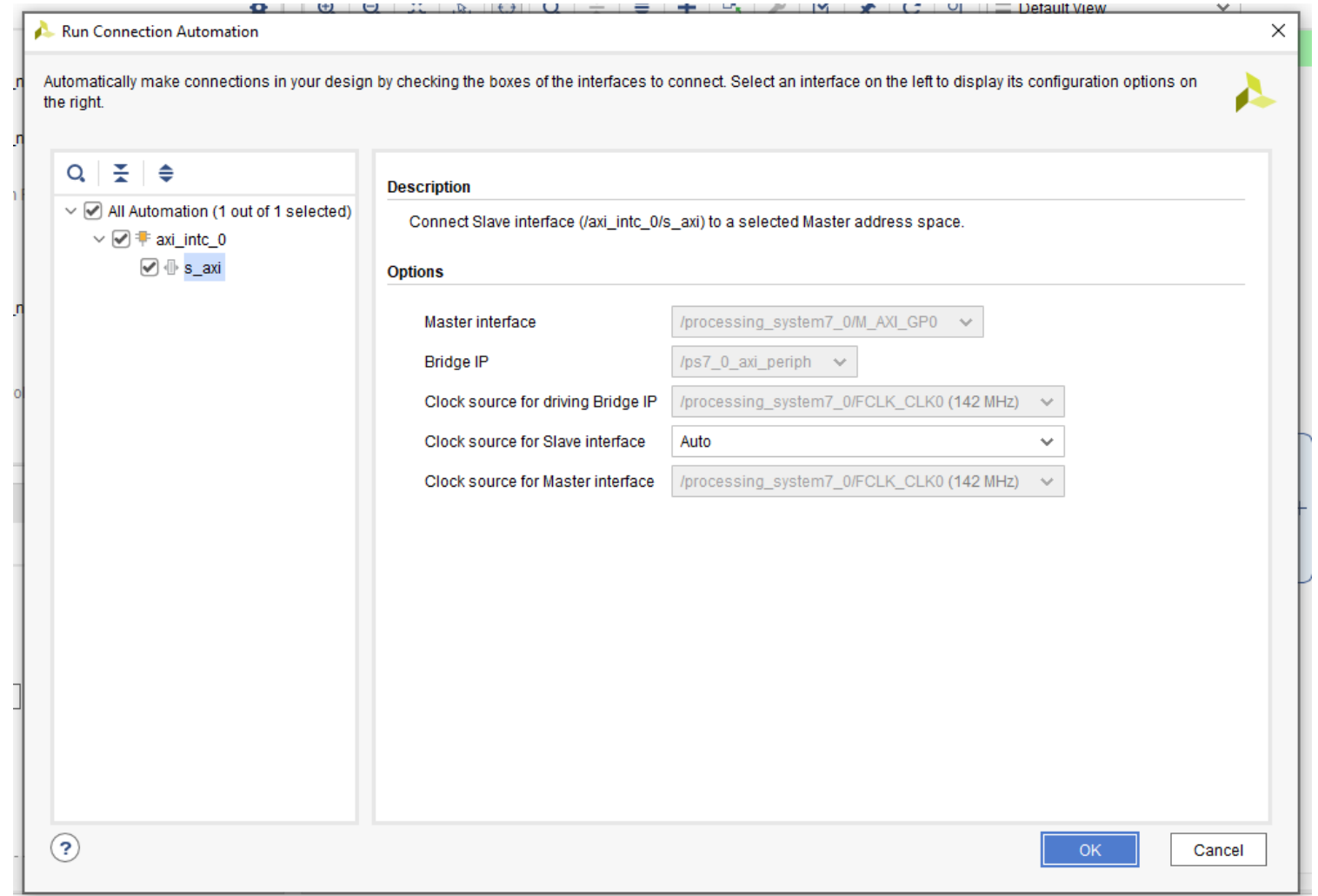
# Lab: FFT Verification

Click on + and add in a AXI  
Interrupt Controller



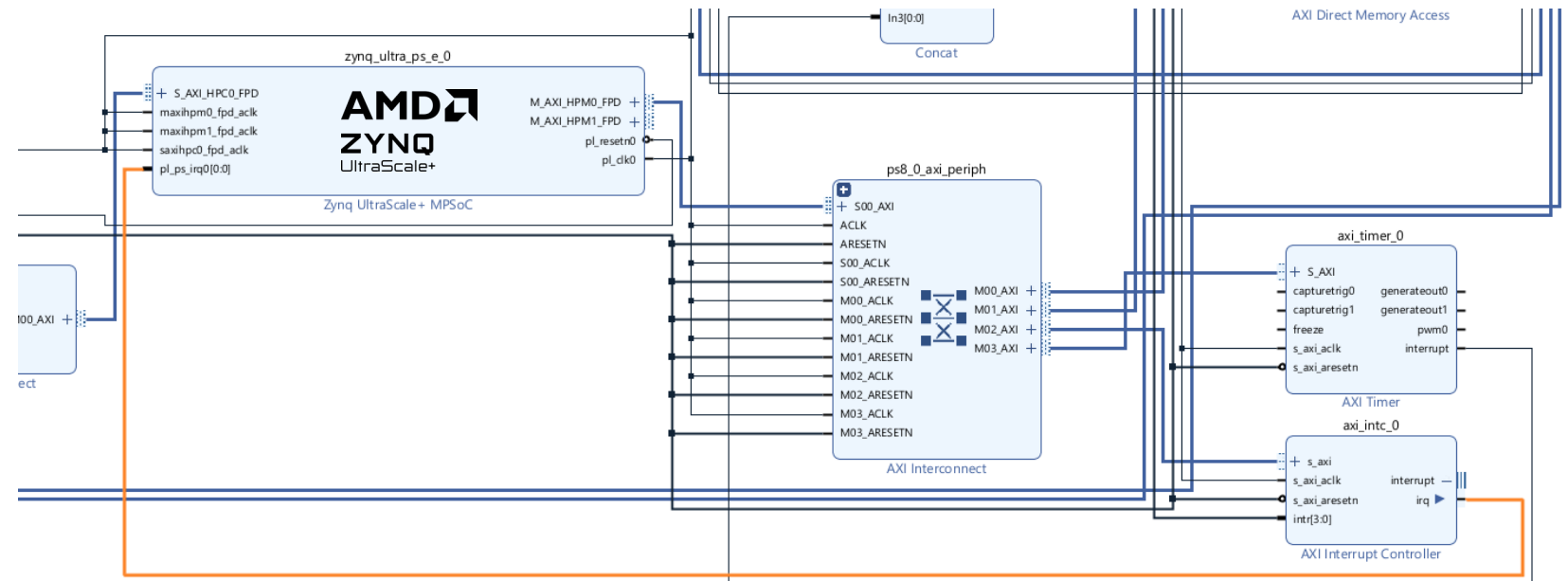
# Lab: FFT Verification

Run the connection  
automation

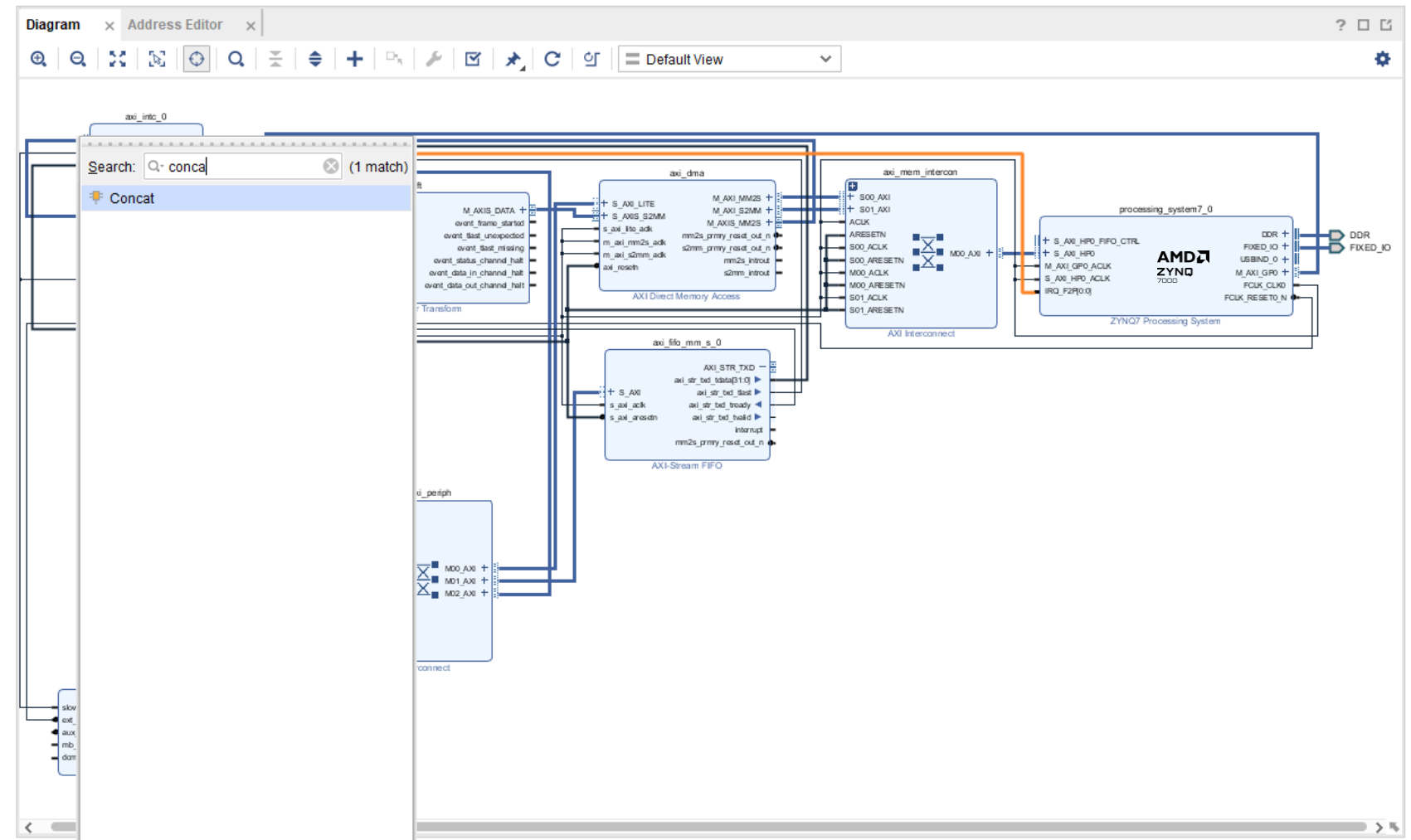


# Lab: FFT Verification

Connect the IRQ output  
from the AXI Interrupt  
Controller to the PL PS  
IRQ port on the Zynq



Click on + and add in a  
concat block





# Lab: FFT Verification

Re-customize IP

**Concat (2.1)**

Documentation IP Location

☐ Show disabled ports

Component Name

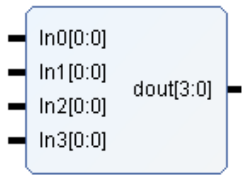
Number of Ports  [1 - 32]

<input type="text" value="AUTO"/>	In0 Width	<input type="text" value="1"/>	[1 - 4096]
<input type="text" value="AUTO"/>	In1 Width	<input type="text" value="1"/>	[1 - 4096]
<input type="text" value="AUTO"/>	In2 Width	<input type="text" value="1"/>	[1 - 4096]
<input type="text" value="AUTO"/>	In3 Width	<input type="text" value="1"/>	[1 - 4096]

Dout Width (Auto)

NOTE: The In0 port is connected to the LSB bits of the output, and the In[Number of Ports - 1] input port is connected to the MSB bits of the output.

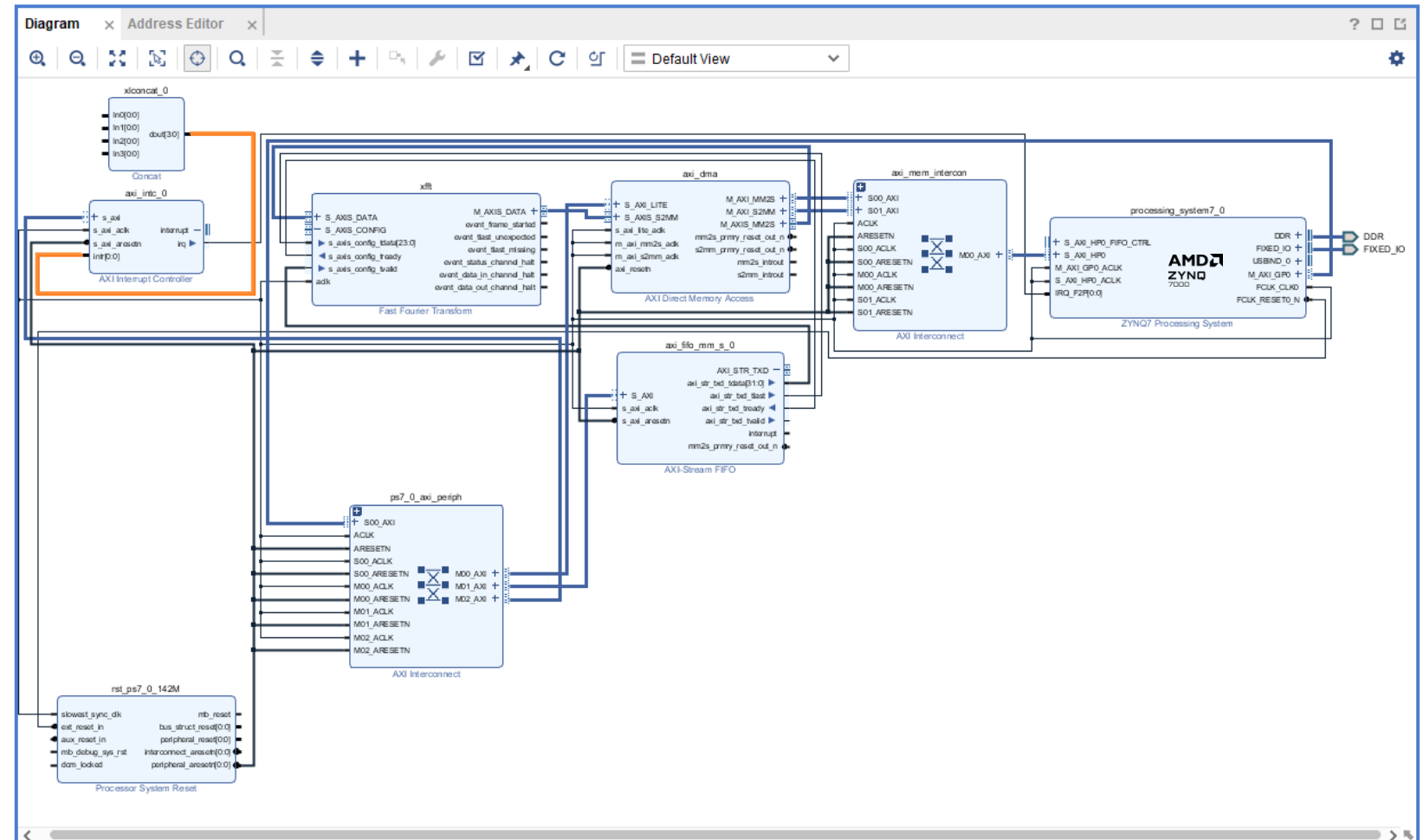
OK Cancel



```
graph LR; In0[In0[0:0]] --- Concat[Concat]; In1[In1[0:0]] --- Concat; In2[In2[0:0]] --- Concat; In3[In3[0:0]] --- Concat; Concat --> dout[dout[3:0]]
```

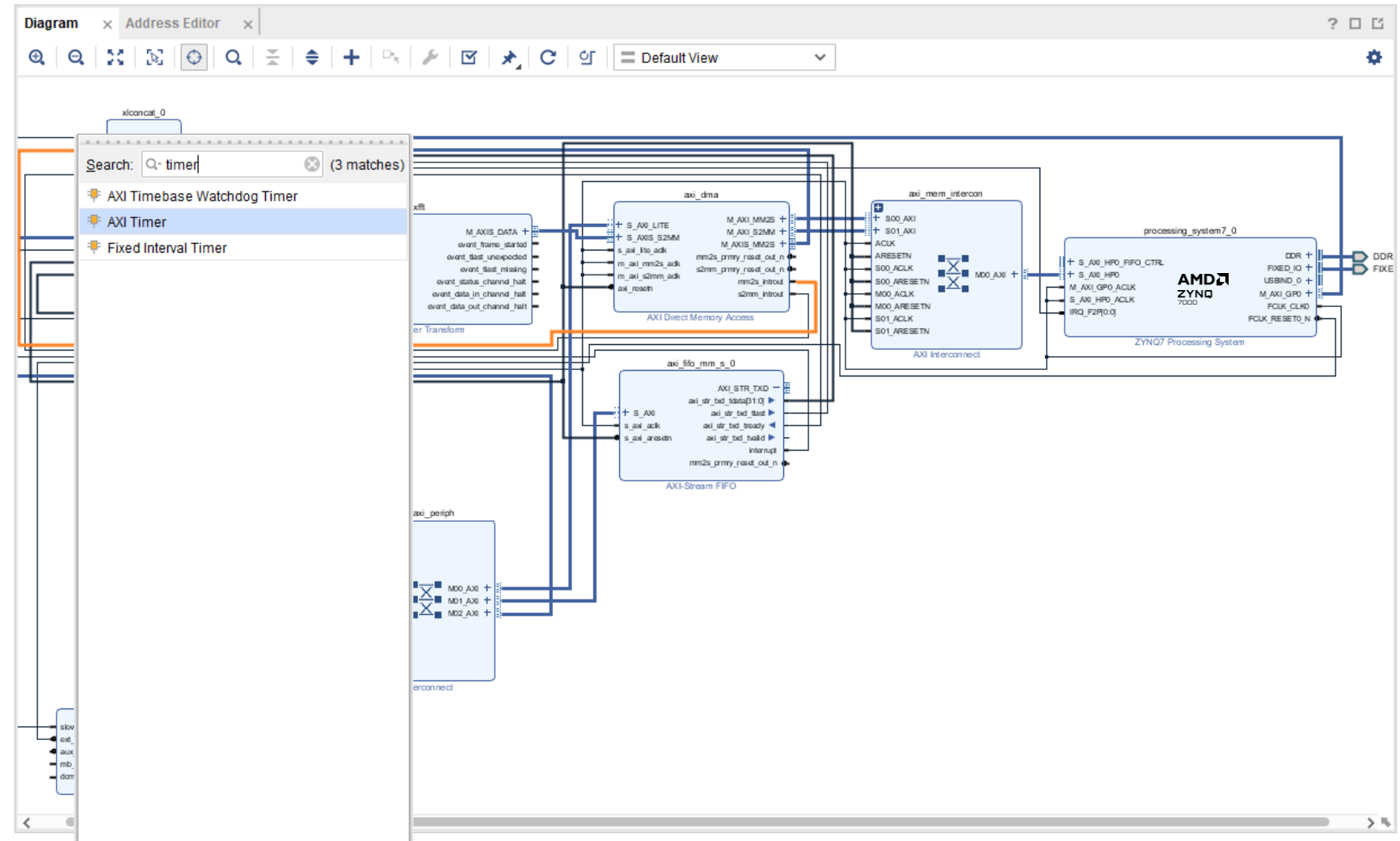
# Lab: FFT Verification

Connect the output of the  
concat block to the AXI  
Interrupt controller INT  
input



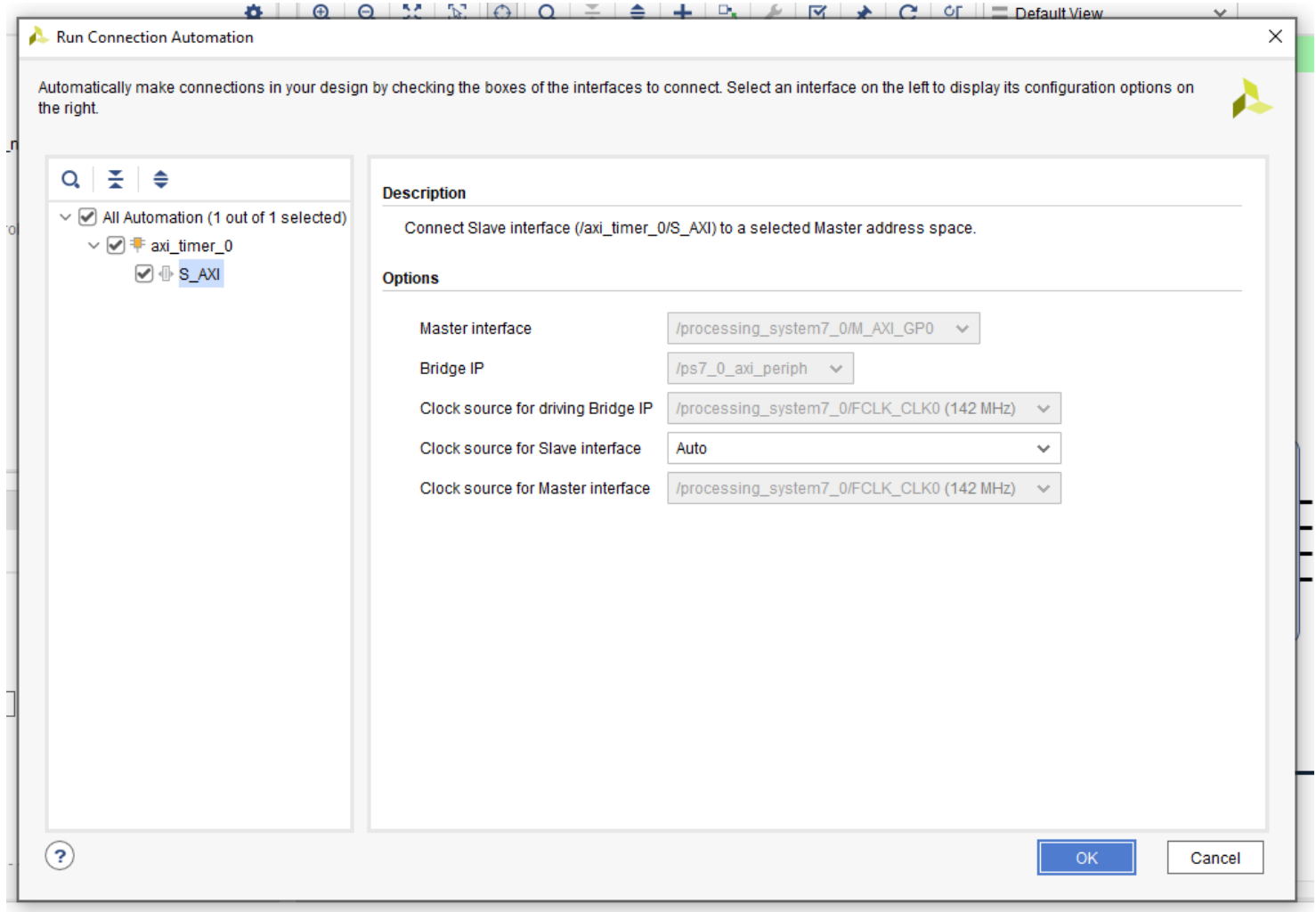
# Lab: FFT Verification

Click on the + symbol  
and add in a AXI Timer



# Lab: FFT Verification

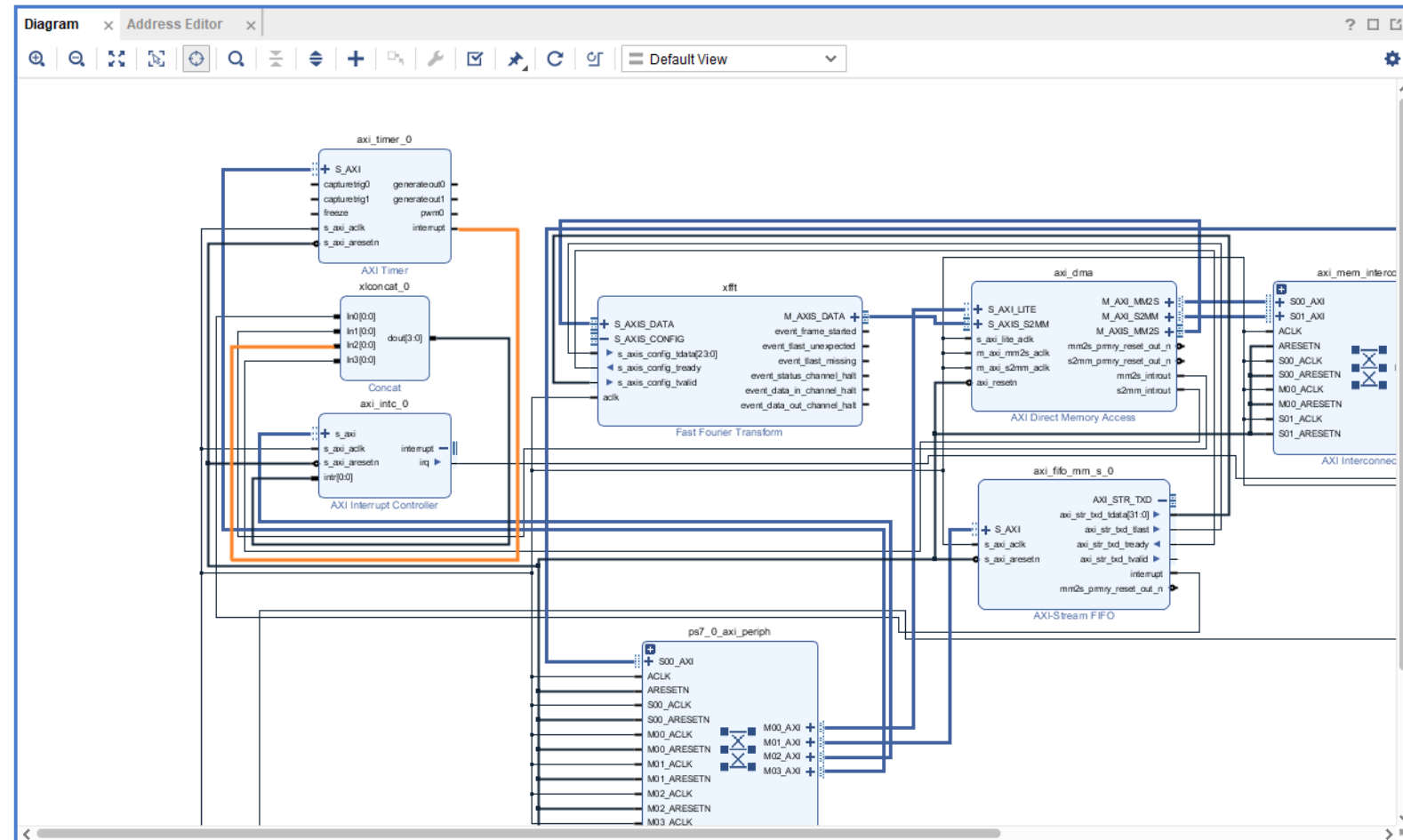
Run the connection automation



# Lab: FFT Verification

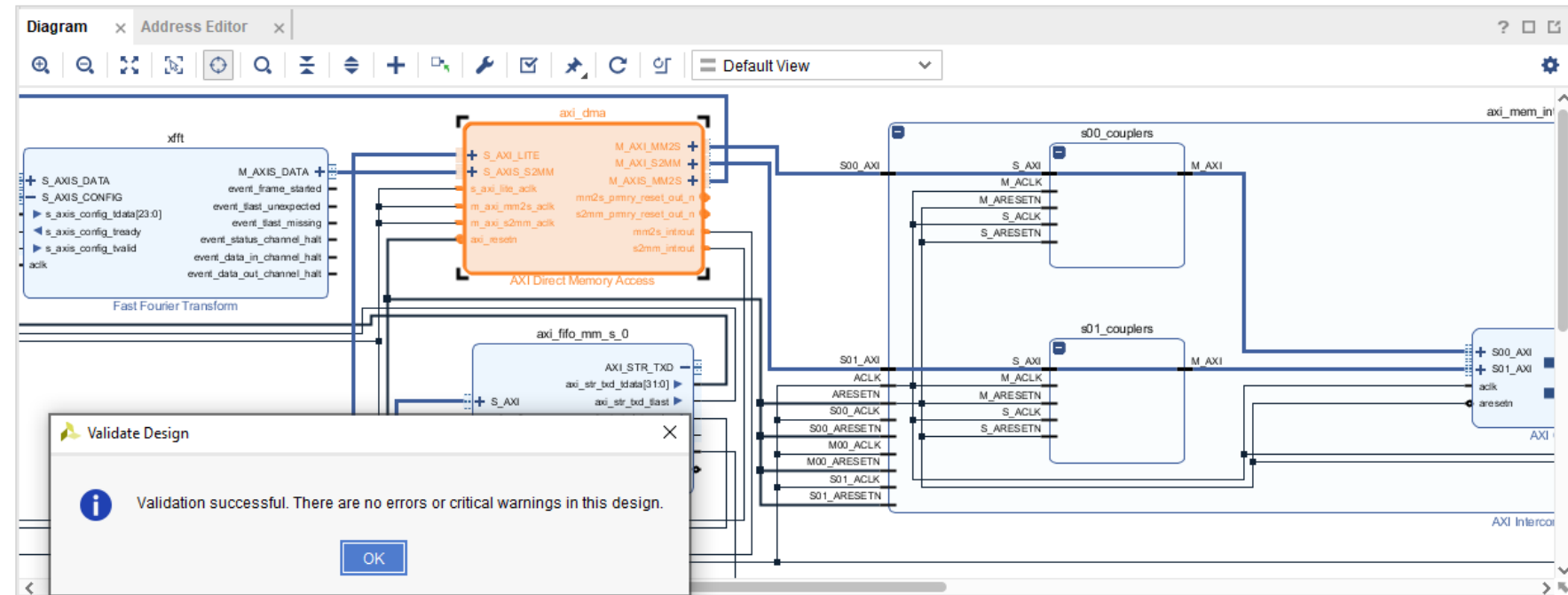
Connect the interrupts to the concat block

- AXI Timer
- AXI DMA MM2S\_INTOUT
- AXI DMA S2MM\_INTOUT
- AXI Stream FIFO



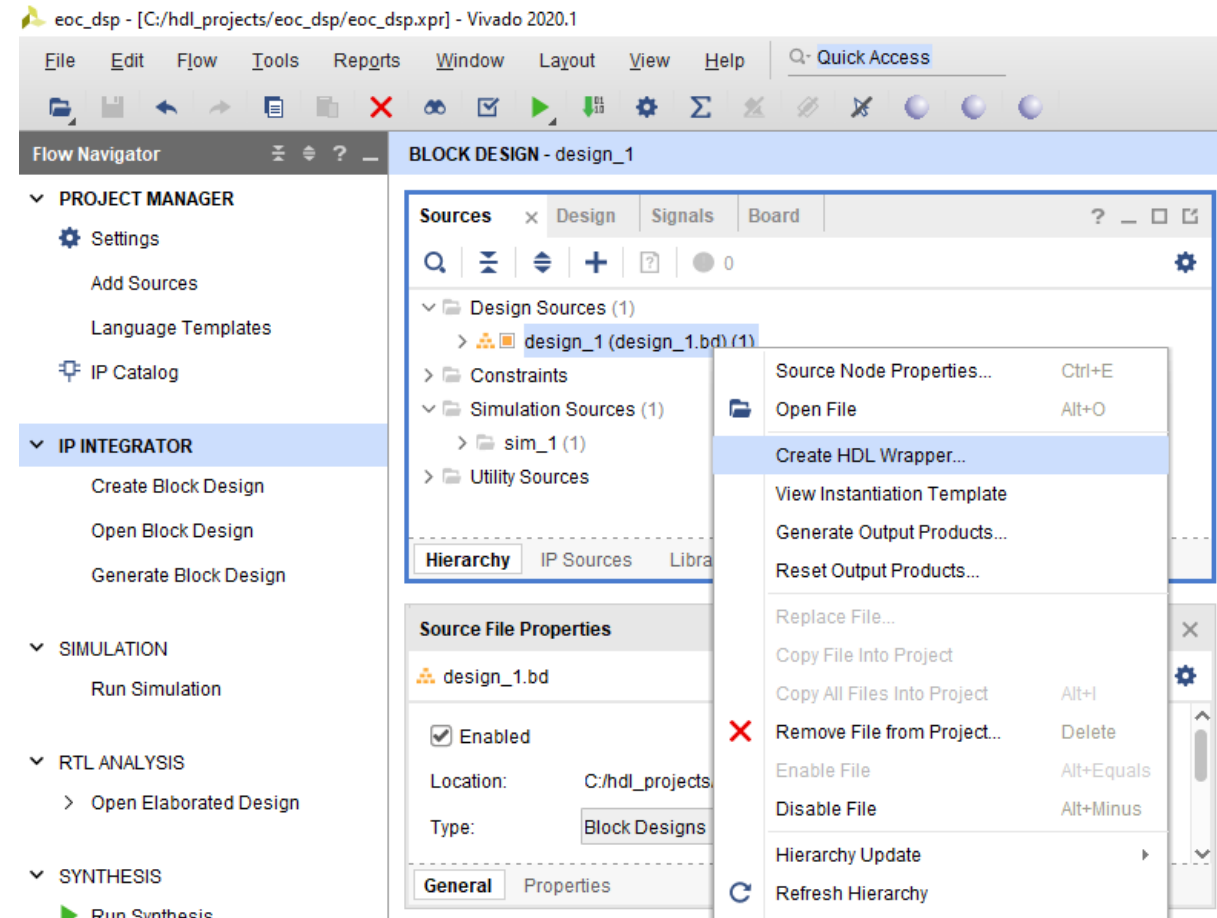
# Lab: FFT Verification

Validate the design  
there should be no  
error or critical  
warnings



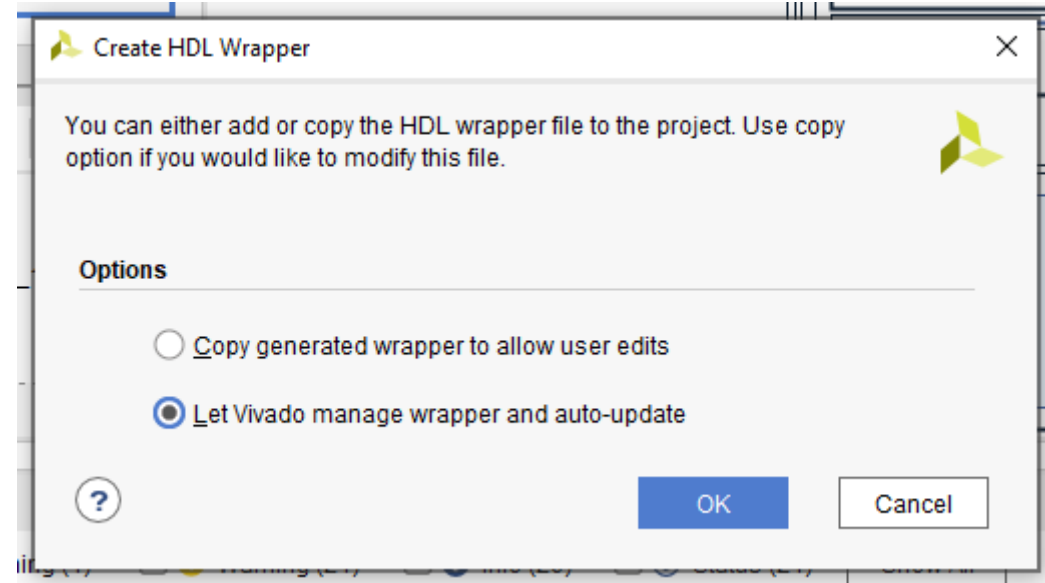
# Lab: FFT Verification

Right click on the design and  
select Create HDL Wrapper



# Lab: FFT Verification

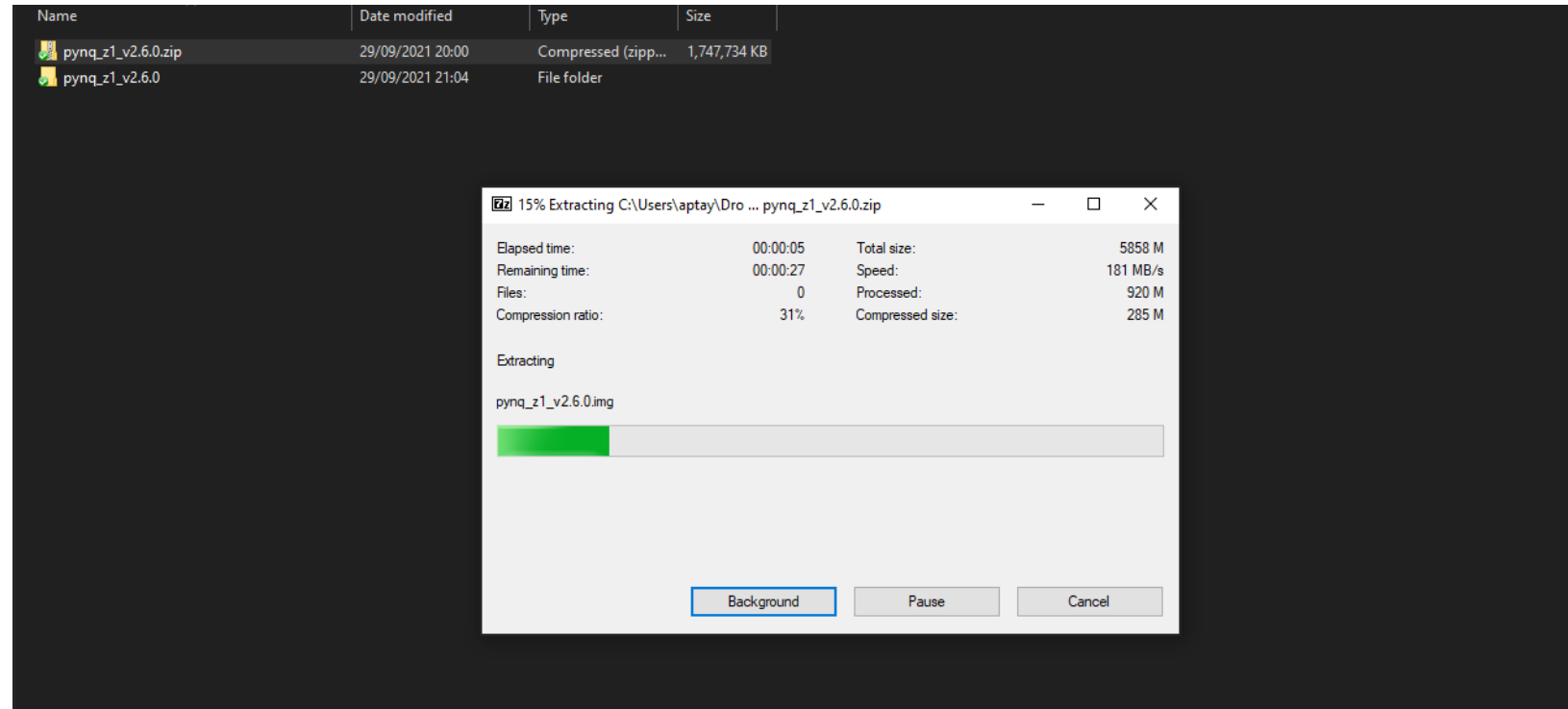
Let Vivado™ manage the wrapper





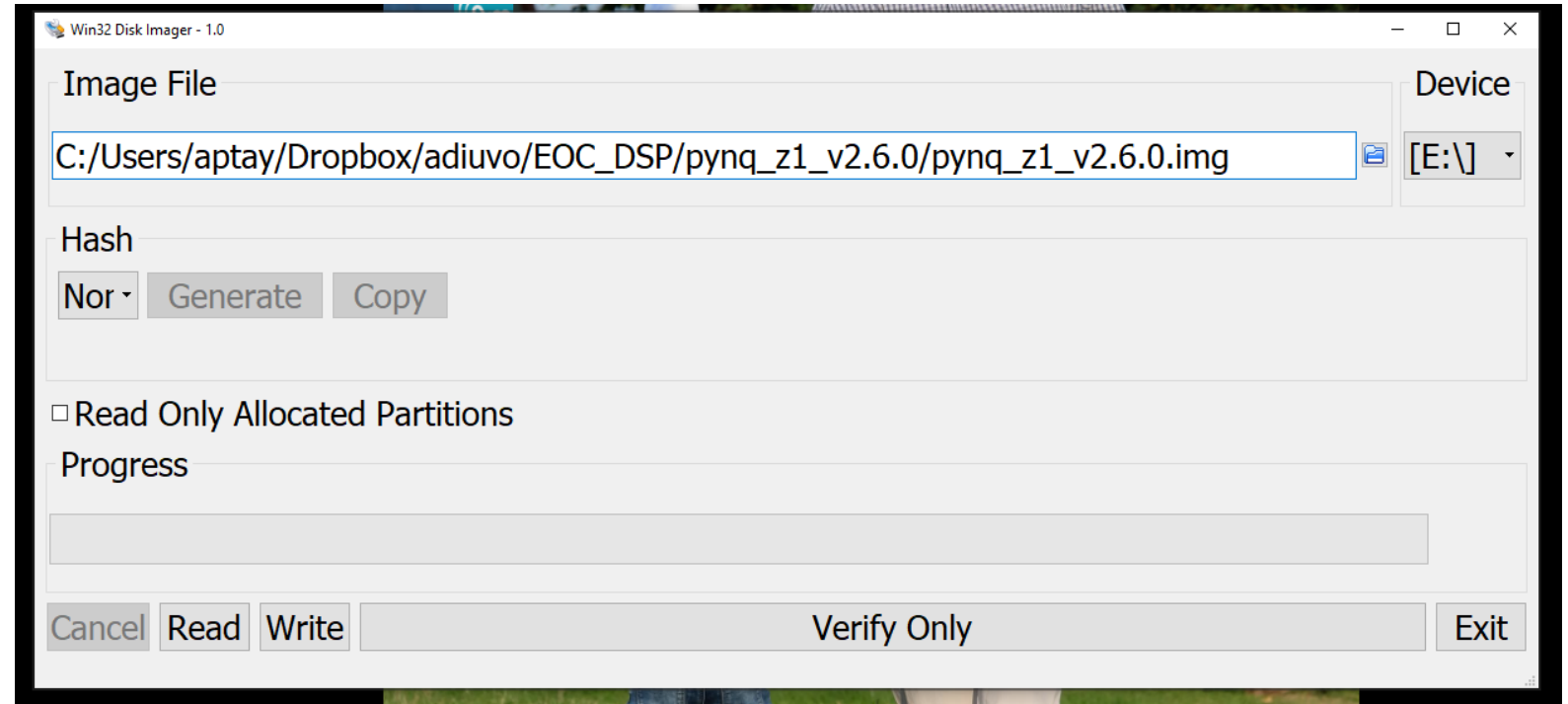
# Lab: FFT Verification

Extract the  
downloaded PYNQ™  
image



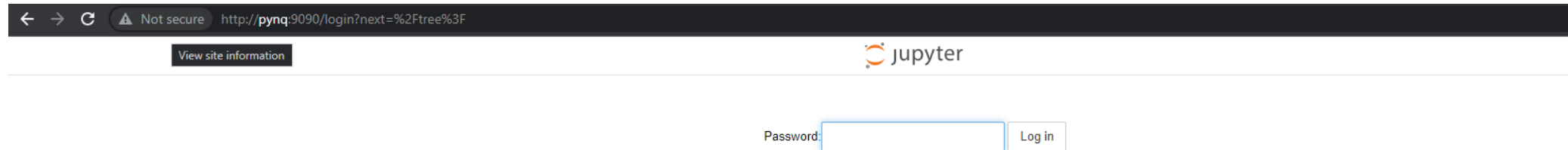
# Lab: FFT Verification

Write the image to a SD card. Once completed, insert the SD card in the Avnet ZU Board. Connect an Ethernet cable and power via a USB cable



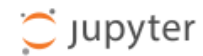
# Lab: FFT Verification

Once the board boots, wait for the LEDs to flash. In a browser enter the address `pynq:9090` when prompted enter the password `xilinx`



# Lab: FFT Verification

Once logged in, you should see the folder structure below

[Logout](#)[Files](#)[Running](#)[Clusters](#)[Nbextensions](#)

Select items to perform actions on them.

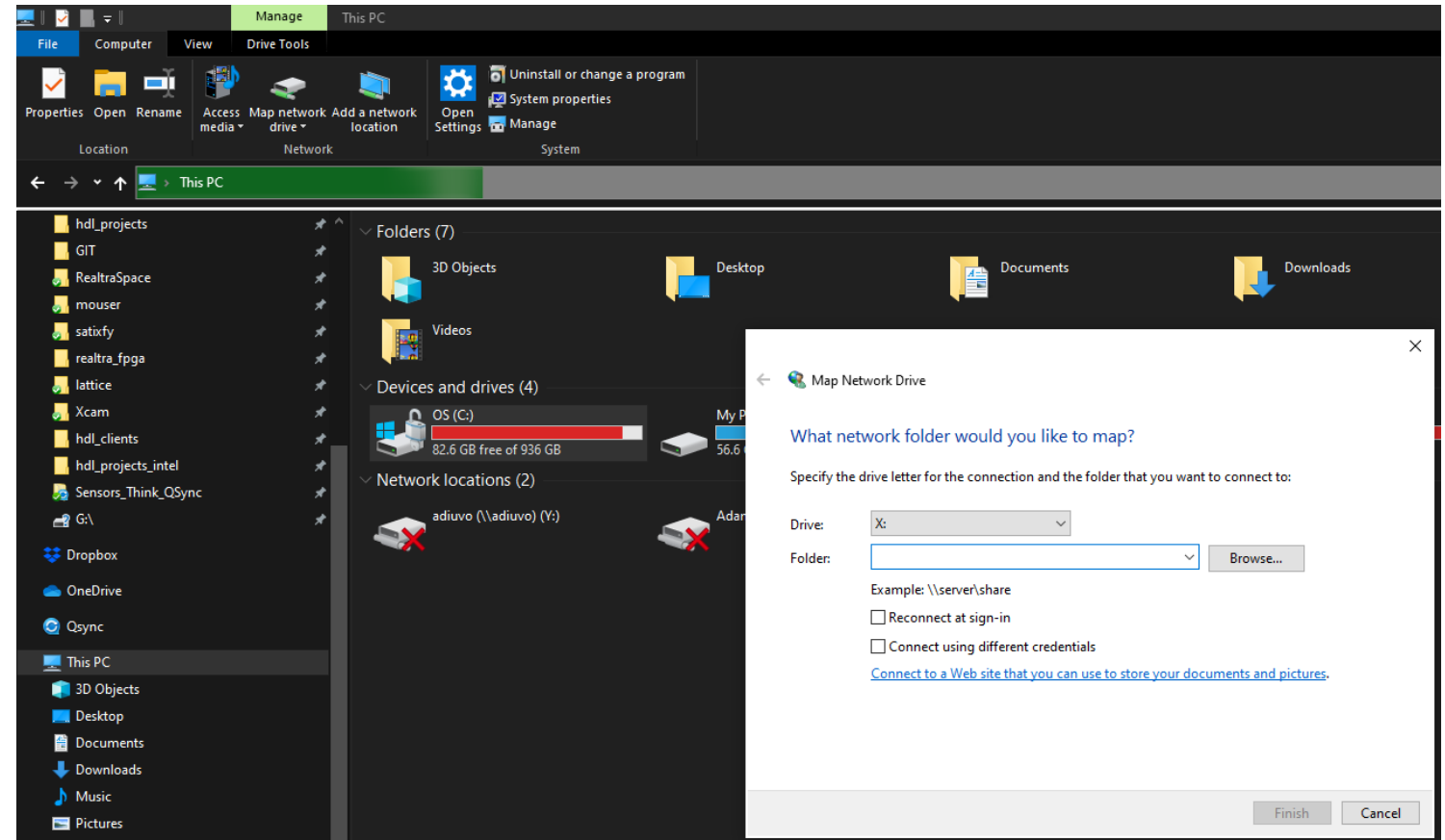
[Upload](#)[New ▾](#)☐ 0 ▾[Name ▾](#)[Last Modified](#)☐[base](#)[a year ago](#)☐[common](#)[a year ago](#)☐[getting\\_started](#)[a year ago](#)☐[logictools](#)[a year ago](#)☐[Welcome to Pynq.ipynb](#)[a year ago](#)

# Lab: FFT Verification

In a file explorer map a  
network drive to

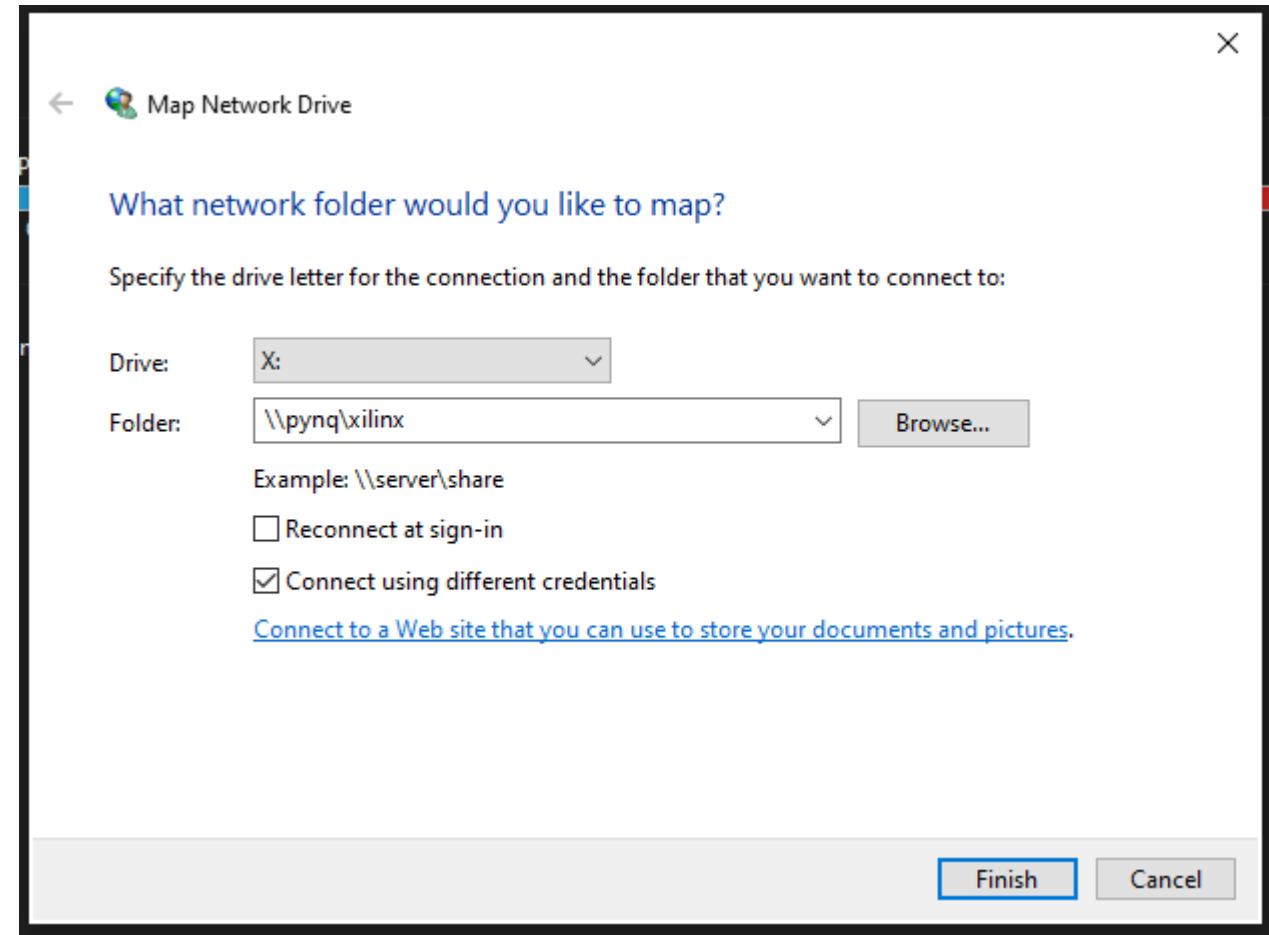
`\\pynq \xilinx`

Select connect using different  
credentials



# Lab: FFT Verification

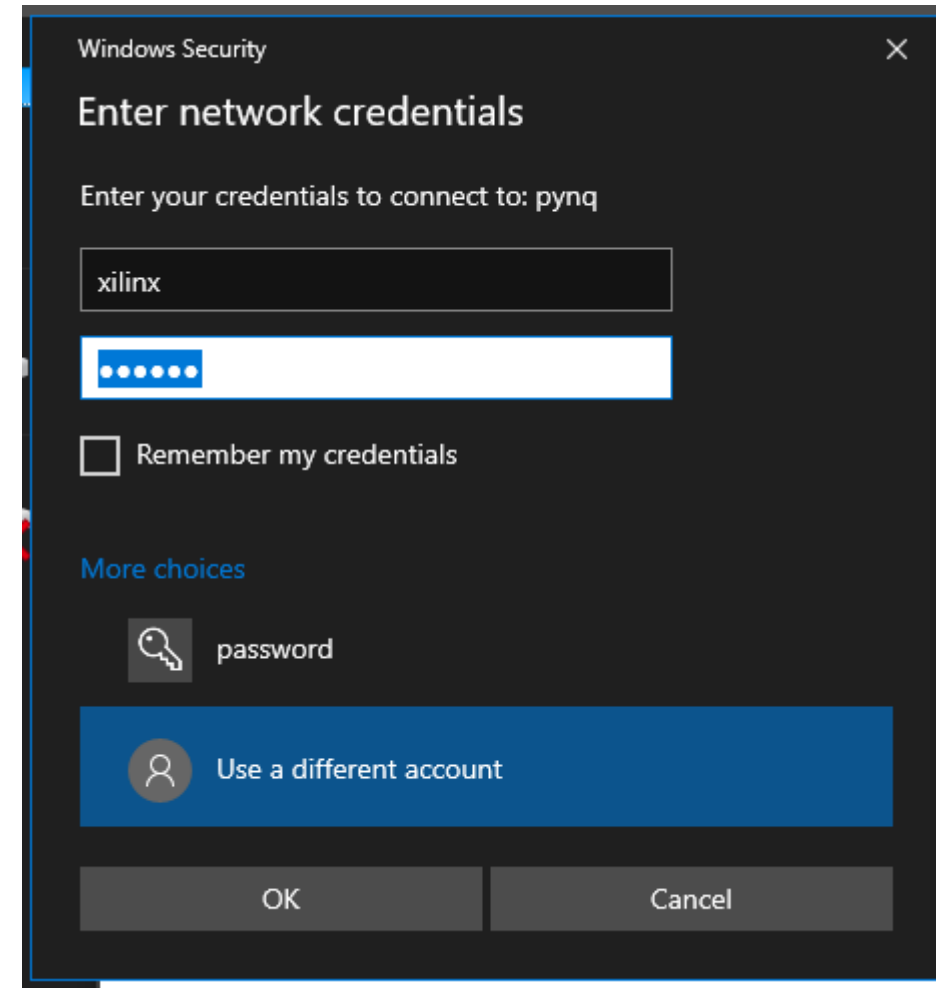
Completed Map drive, click OK



# Lab: FFT Verification

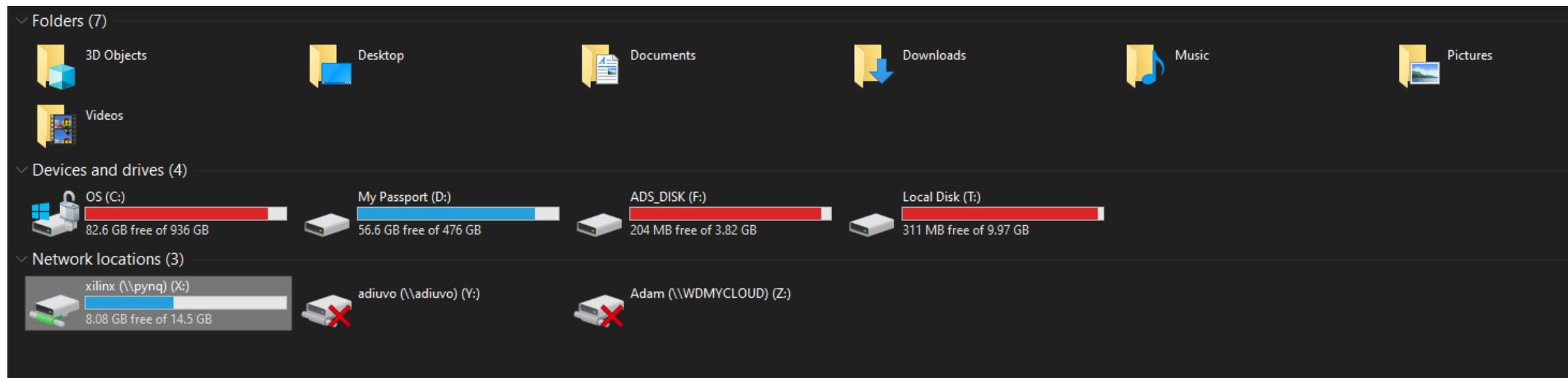
Enter the username and password as

Xilinx click OK



# Lab: FFT Verification

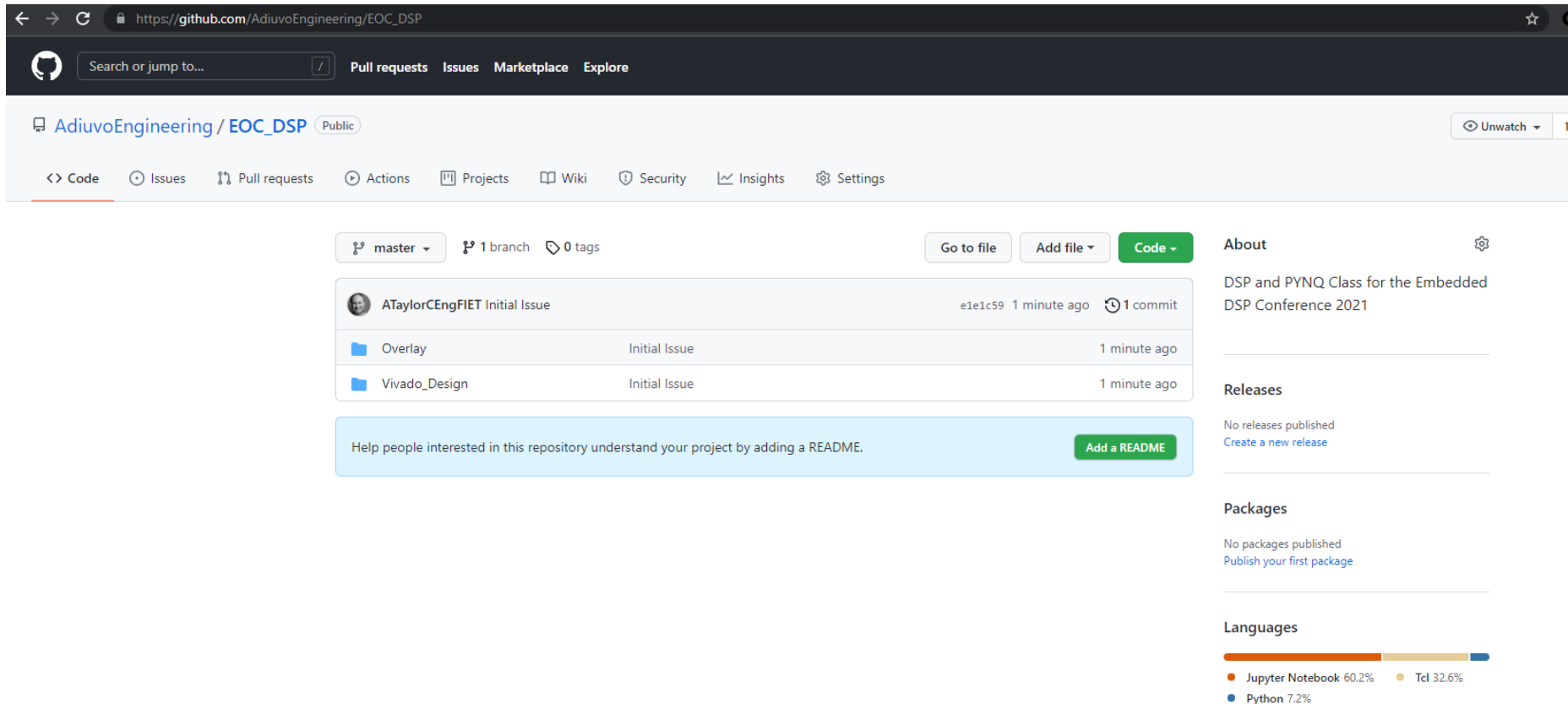
The PYNQ™ drive should not appear as a samba server





# Lab: FFT Verification

Clone the repository - <https://github.com/ATaylorCEngFIET/MZ490>

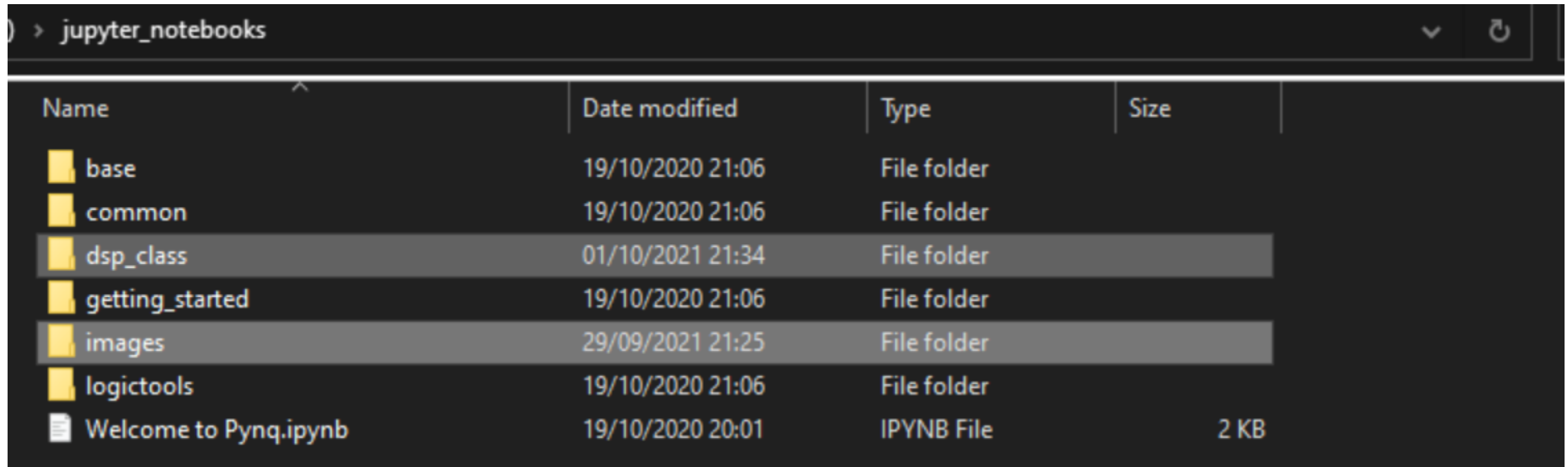


The screenshot shows the GitHub repository page for `AdiuvoEngineering / EOC_DSP`. The repository is public and has 1 branch (master) and 0 tags. The commit history shows a single commit by ATaylorCEngFIET, titled "Initial Issue", with two files added: `Overlay` and `Vivado_Design`. The repository description is "DSP and PYNQ Class for the Embedded DSP Conference 2021". The repository also includes a section for "About", "Releases", "Packages", and "Languages". The "Languages" section shows a bar chart with the following data:

Language	Percentage
Jupyter Notebook	60.2%
Tcl	32.6%
Python	7.2%

# Lab: FFT Verification

From the Cloned Repo copy the directory Images and dsp\_class to the PYNQ™ boards Jupyter notebooks directory



Name	Date modified	Type	Size
base	19/10/2020 21:06	File folder	
common	19/10/2020 21:06	File folder	
dsp_class	01/10/2021 21:34	File folder	
getting_started	19/10/2020 21:06	File folder	
images	29/09/2021 21:25	File folder	
logictools	19/10/2020 21:06	File folder	
Welcome to Pynq.ipynb	19/10/2020 20:01	IPYNB File	2 KB

# Lab: FFT Verification

You should see a new directory in the PYNQ™ environment. Select DSP\_CLASS

[Logout](#)[Files](#)[Running](#)[Clusters](#)[Nbextensions](#)

Select items to perform actions on them.

[Upload](#)[New ▾](#)

<input type="checkbox"/> 0 ▾ 		Name ▾	Last Modified
<input type="checkbox"/>	base		a year ago
<input type="checkbox"/>	common		a year ago
<input type="checkbox"/>	dsp_class		16 hours ago
<input type="checkbox"/>	getting_started		a year ago
<input type="checkbox"/>	images		3 days ago
<input type="checkbox"/>	logictools		a year ago
<input type="checkbox"/>	 Welcome to Pynq.ipynb		a year ago

# Lab: FFT Verification

Select fft.ipynb it will open and start running



jupyter Logout

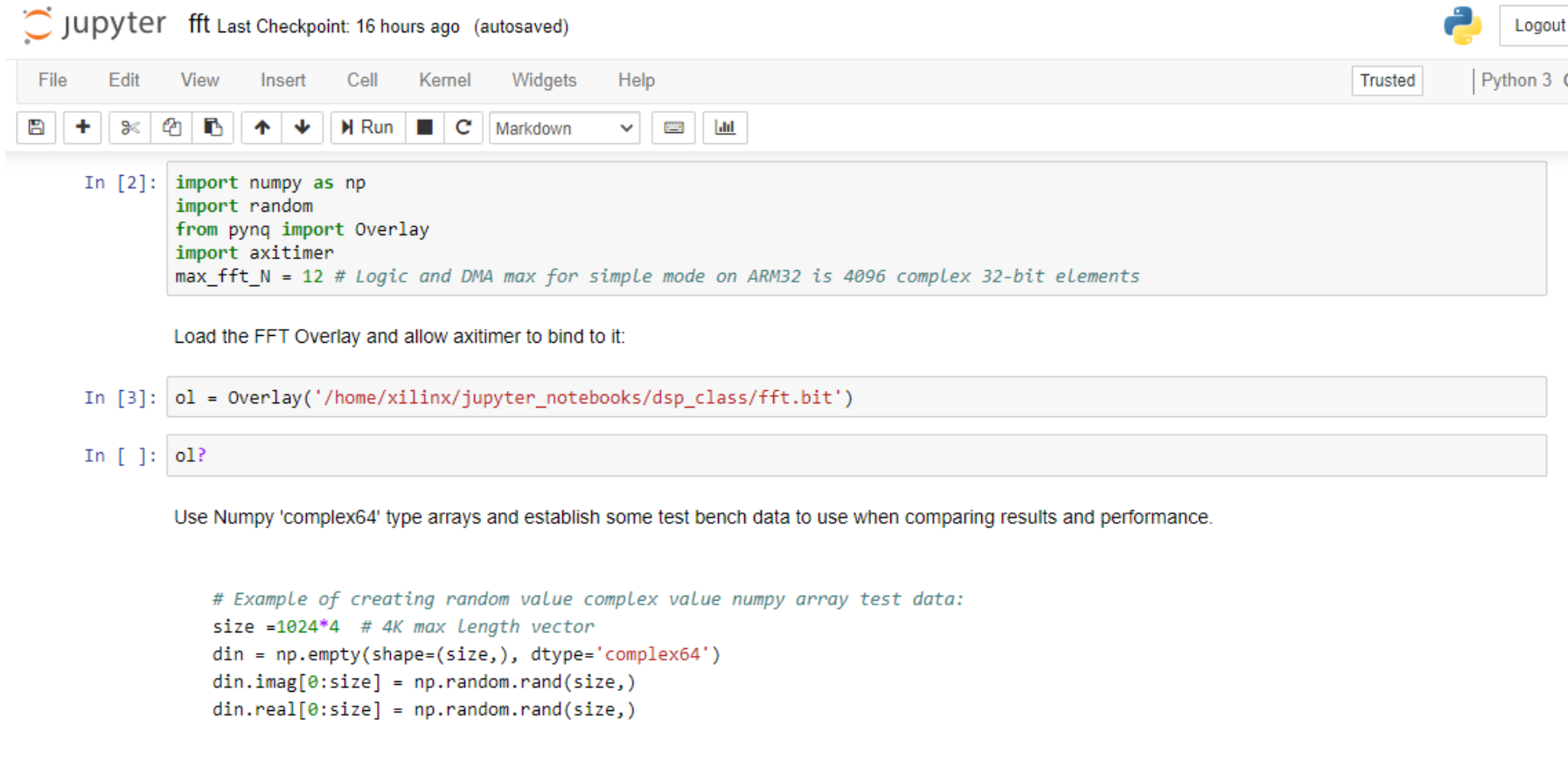
Files Running Clusters Nbextensions

Select items to perform actions on them. Upload New Refresh

<input type="checkbox"/> 0	<input type="checkbox"/> / dsp_class	Name	Last Modified
	..		seconds ago
<input type="checkbox"/>	fft.ipynb		Running 14 hours ago
<input type="checkbox"/>	axidma.py		7 months ago
<input type="checkbox"/>	axififo.py		7 months ago
<input type="checkbox"/>	axitimer.py		7 months ago
<input type="checkbox"/>	fft.bit		15 hours ago
<input type="checkbox"/>	fft.hwh		15 hours ago
<input type="checkbox"/>	sds_trace_data.dat		3 days ago

# Lab: FFT Verification

Run each cell in turn in the notebook and notice the difference in performance between SW and HW Implementations



The image shows a Jupyter Notebook interface with the following content:

```
jupyter fft Last Checkpoint: 16 hours ago (autosaved)
```

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

In [2]:

```
import numpy as np
import random
from pyng import Overlay
import axitimer
max_fft_N = 12 # Logic and DMA max for simple mode on ARM32 is 4096 complex 32-bit elements
```

Load the FFT Overlay and allow axitimer to bind to it:

In [3]:

```
ol = Overlay('/home/xilinx/jupyter_notebooks/dsp_class/fft.bit')
```

In [ ]:

```
ol?
```

Use Numpy 'complex64' type arrays and establish some test bench data to use when comparing results and performance.

```
# Example of creating random value complex value numpy array test data:
size = 1024*4 # 4K max length vector
din = np.empty(shape=(size,), dtype='complex64')
din.imag[0:size] = np.random.rand(size,)
din.real[0:size] = np.random.rand(size,)
```



AMD sponsored this workshop, including engineering hours. AMD, and the AMD Arrow logo, PYNQ, UltraScale+, Vitis, Vivado, Zynq, and combinations thereof are trademarks of Advanced Micro Devices, Inc. Other product names used in this publication are for identification purposes only and may be trademarks of their respective companies.



ADIUVO  
ENGINEERING AND TRAINING, LTD.

[www.adiuvoengineering.com](http://www.adiuvoengineering.com)



[adam@adiuvoengineering.com](mailto:adam@adiuvoengineering.com)