



Signal Processing with FPGA, Python & no RTL Design!

Adam Taylor

Adam@AdiuvoEngineering.com

Lab : FFT Acceleration

Open a browser and go to

www.pynq.io



What is PYNQ?

PYNQ is an open-source project from Xilinx® that makes it easier to use Xilinx 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

PYNQ Architecture Diagram:

- Applications:** Jupyter/ IPython, PYNQ notebooks, matplotlib, caffe, pytorch, opencv
- Software:** Python, PYNQ libs, MMIO, numpy
- Linux kernel:** GPID, Interrupt, Overlay, PL, allocate, pynq device, Xilinx Runtime (XRT)
- Hardware:** FPGA, Vitis IPs, PYNQ IPs, User IPs

Supported Hardware: ZYNQ, ZYNQ ULTRASCALE+, ZYNQ RFSoC, XILINX ALVEO.

Who is PYNQ for?

PYNQ is intended to be used by a wide range of designers and developers including:

- Software developers who want to take advantage of the capabilities of Xilinx platforms without having to use ASIC-style design tools to design hardware.
- System architects who want an easy software interface and framework for rapid prototyping and development of their Zynq, Alveo and AWS F1 designs.

Lab : FFT Acceleration

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

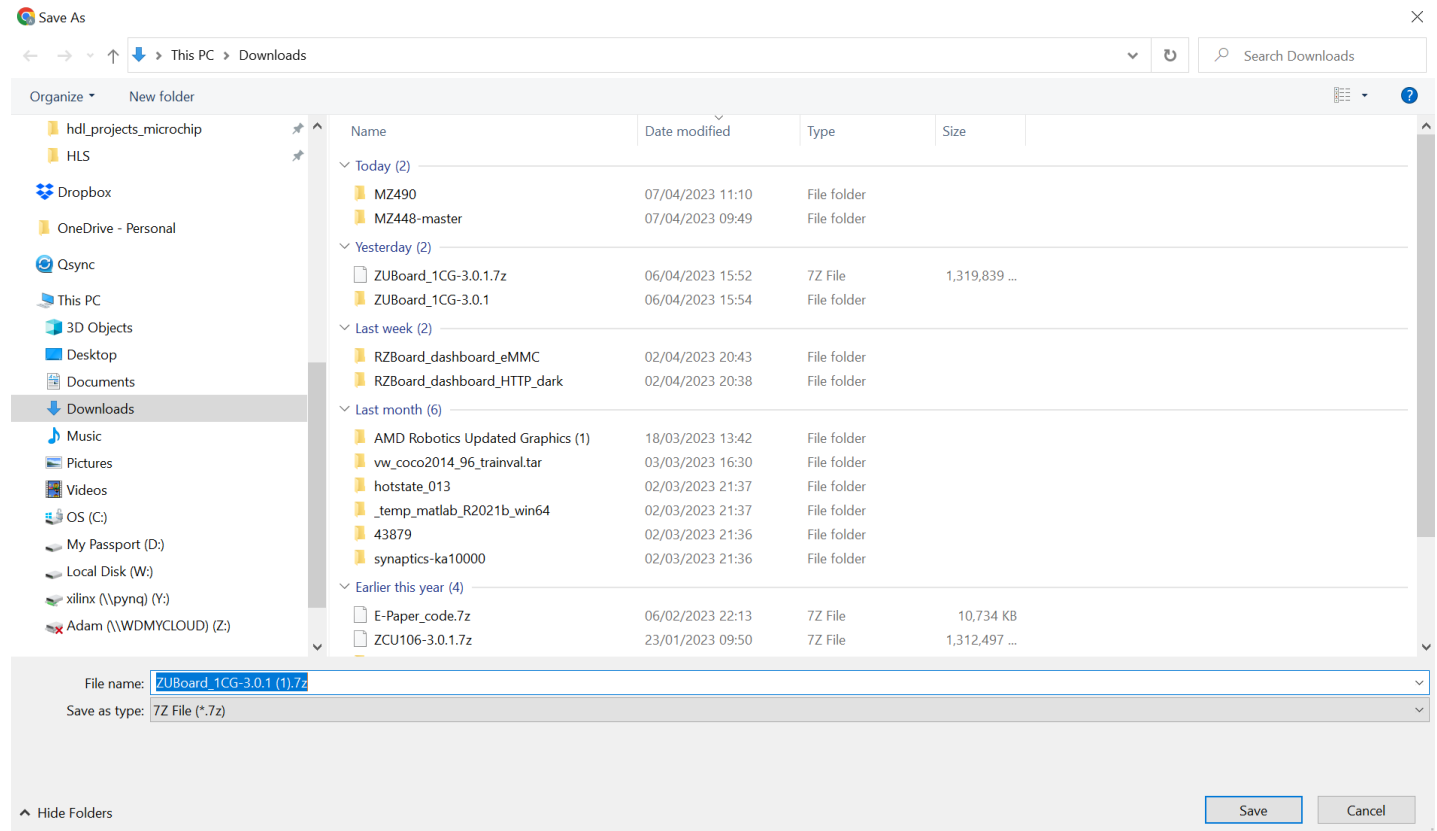
Downloadable PYNQ images

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	v3.0.1	v2.7 v2.6	PYNQ setup guide	TUL Pynq-Z2
PYNQ-Z1	v3.0.1	v2.7 v2.6	PYNQ setup guide	Digilent Pynq-Z1
PYNQ-ZU	v3.0.1	v2.7 v2.6	GitHub project page	TUL PYNQ-ZU
Kria KV260*	Ubuntu 22.04		Kria PYNQ setup	Xilinx Kria KV260
Kria KR260*	Ubuntu 22.04		Kria PYNQ setup	Xilinx Kria KR260
ZCU104	v3.0.1	v2.7 v2.6	PYNQ setup guide	Xilinx ZCU104
RFSoc 2x2	v3.0.1	v2.7 v2.6	RFSoc-PYNQ	XUP RFSoc 2x2
RFSoc 4x2	v3.0.1	v2.7	RFSoc-PYNQ	XUP RFSoc 4x2
ZCU111	v3.0.1	v2.7 v2.6	RFSoc-PYNQ	Xilinx ZCU111
ZCU208	v3.0.1		RFSoc-PYNQ	Xilinx ZCU208
Ultra96V2	v3.0.1	v2.7 v2.6	Avnet PYNQ webpage	Avnet Ultra96V2
Ultra96 (legacy)	v3.0.1	v2.7 v2.6	See Ultra96V2	See Ultra96V2
ZUBoard 1CG	v3.0.1		GitHub project page	Avnet ZUBoard 1CG

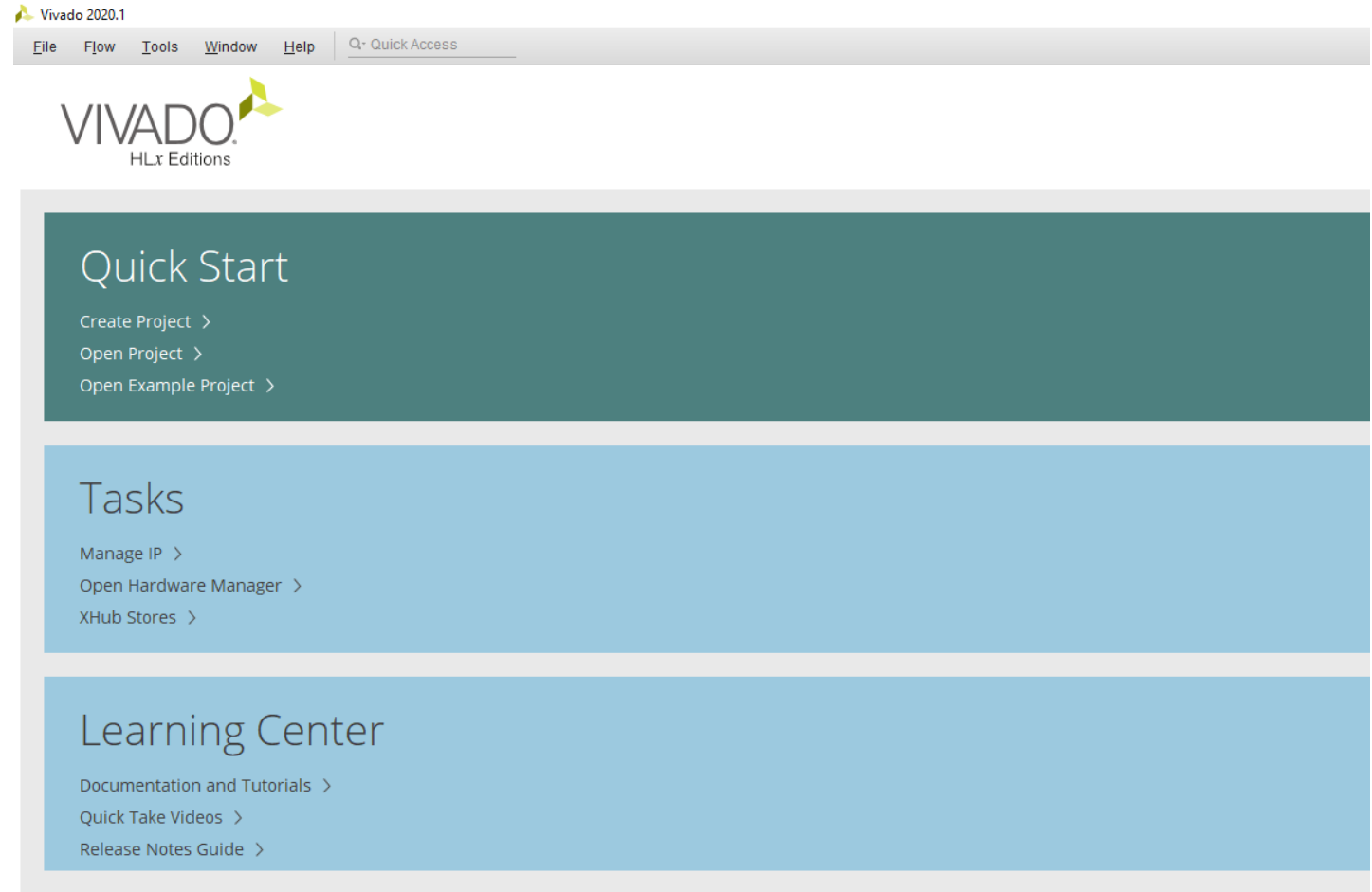
Lab : FFT Acceleration

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



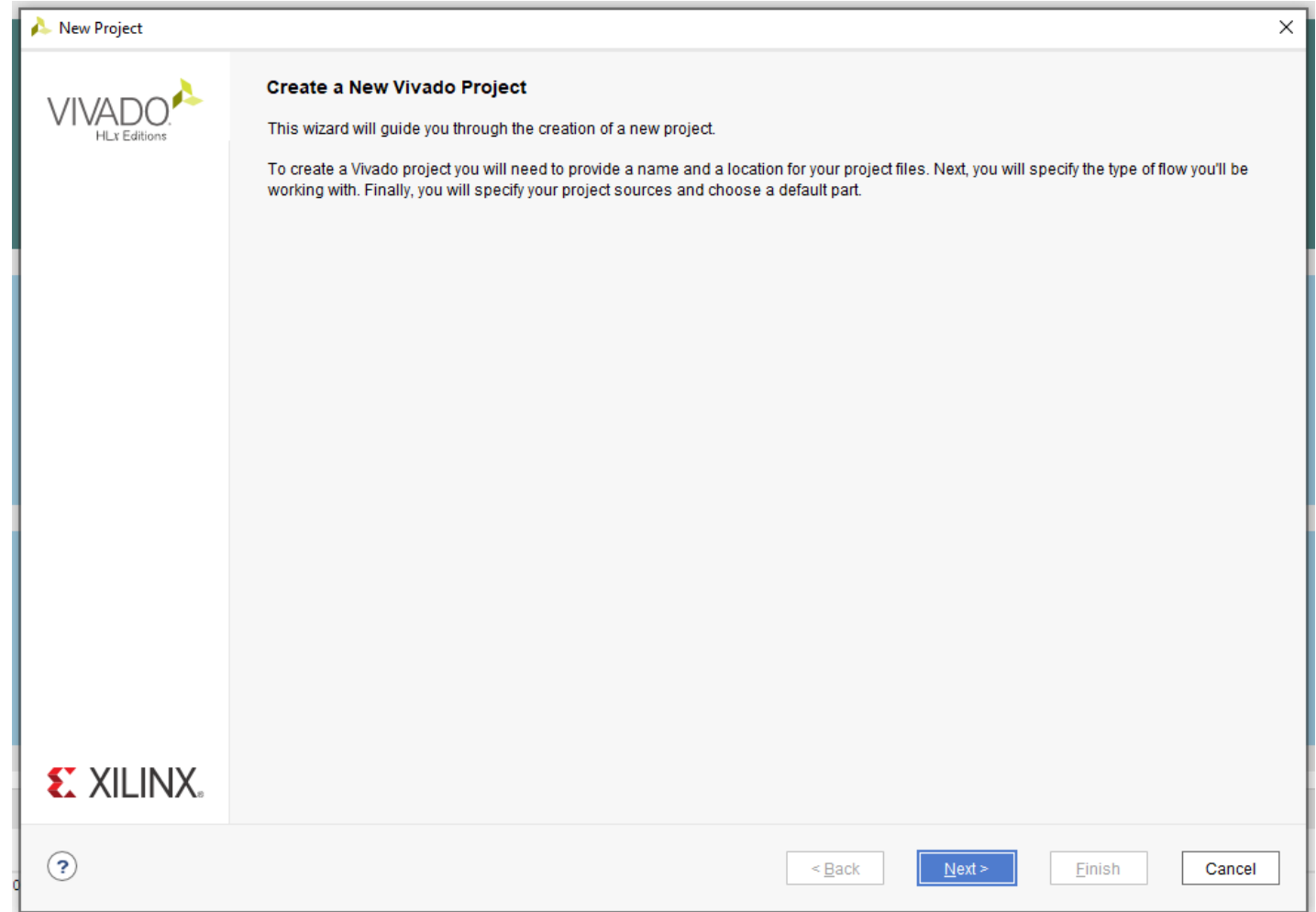
Lab : FFT Acceleration

Open Vivado and select Xhub
Stores



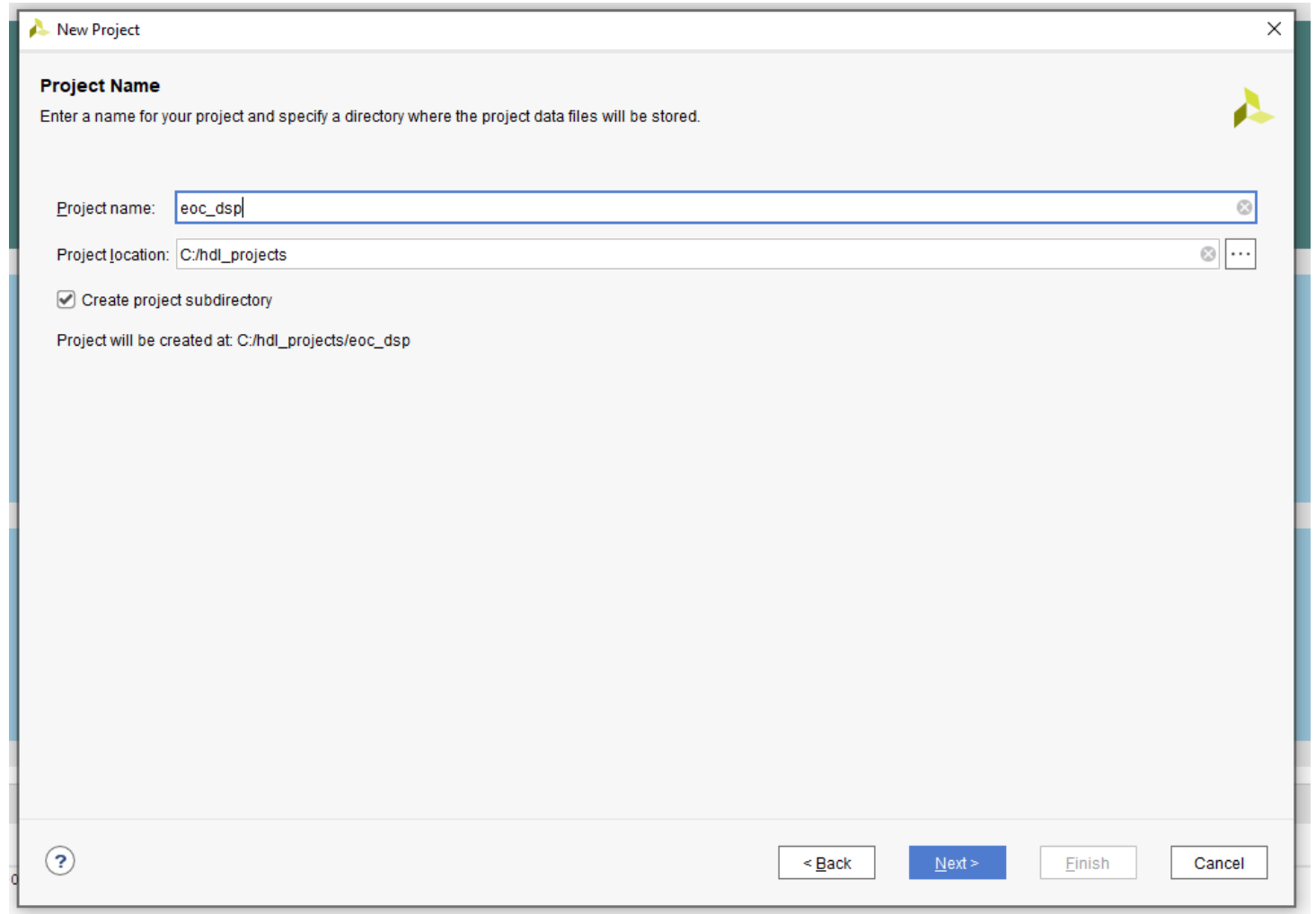
Lab : FFT Acceleration

Create a new project



Lab : FFT Acceleration

Enter a name and location



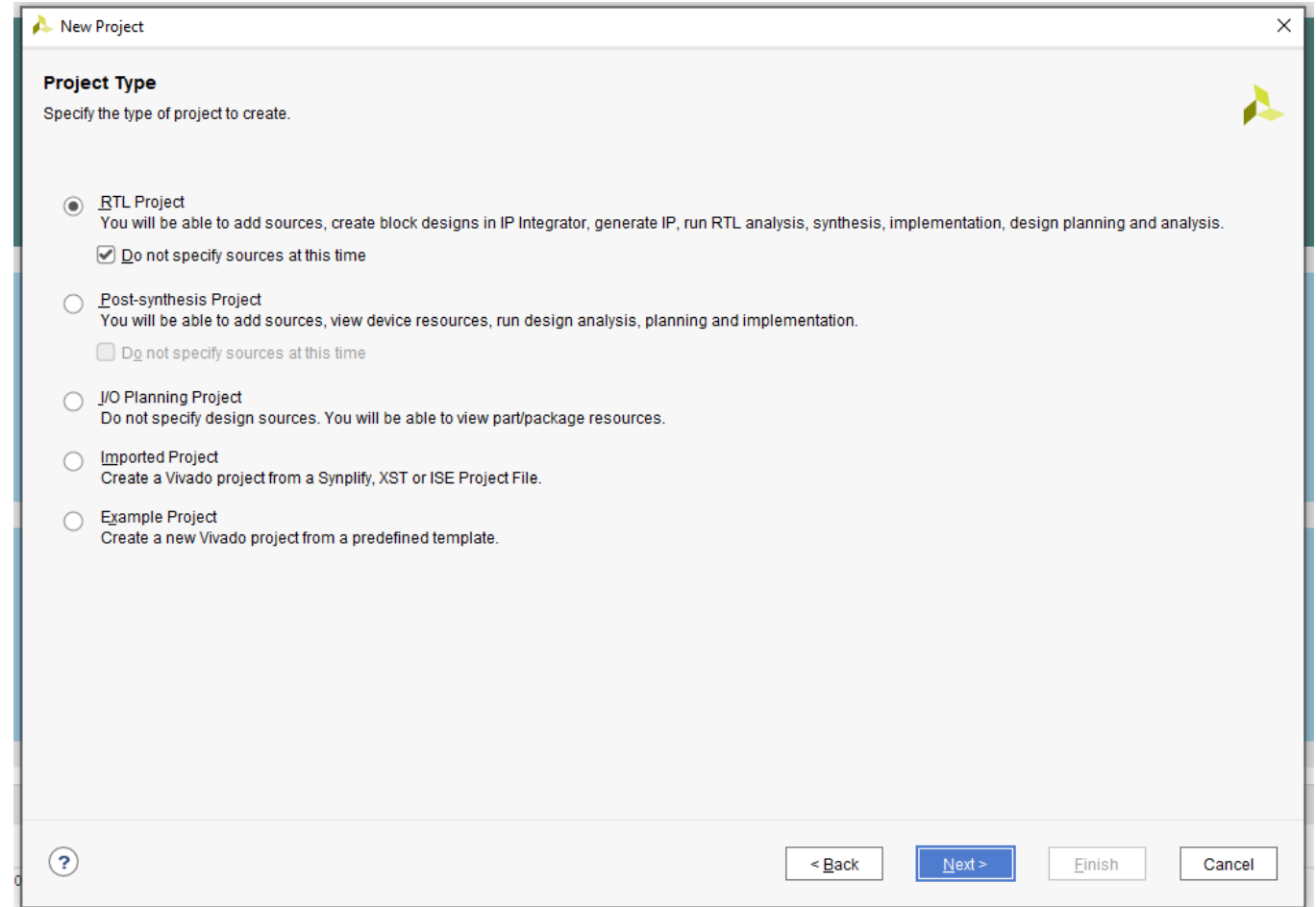
The image shows a 'New Project' dialog box with the following fields and options:

- 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

At the bottom, there are four buttons: ? (help), < Back, Next > (highlighted in blue), Finish, and Cancel.

Lab : FFT Acceleration

Select RTL Project



The image shows a 'New Project' dialog box from a software application. The dialog has a title bar with a yellow Vivado icon and the text 'New Project'. 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 with a 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 Acceleration

Select the ZU Board






 New Project

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

Parts | **Boards**

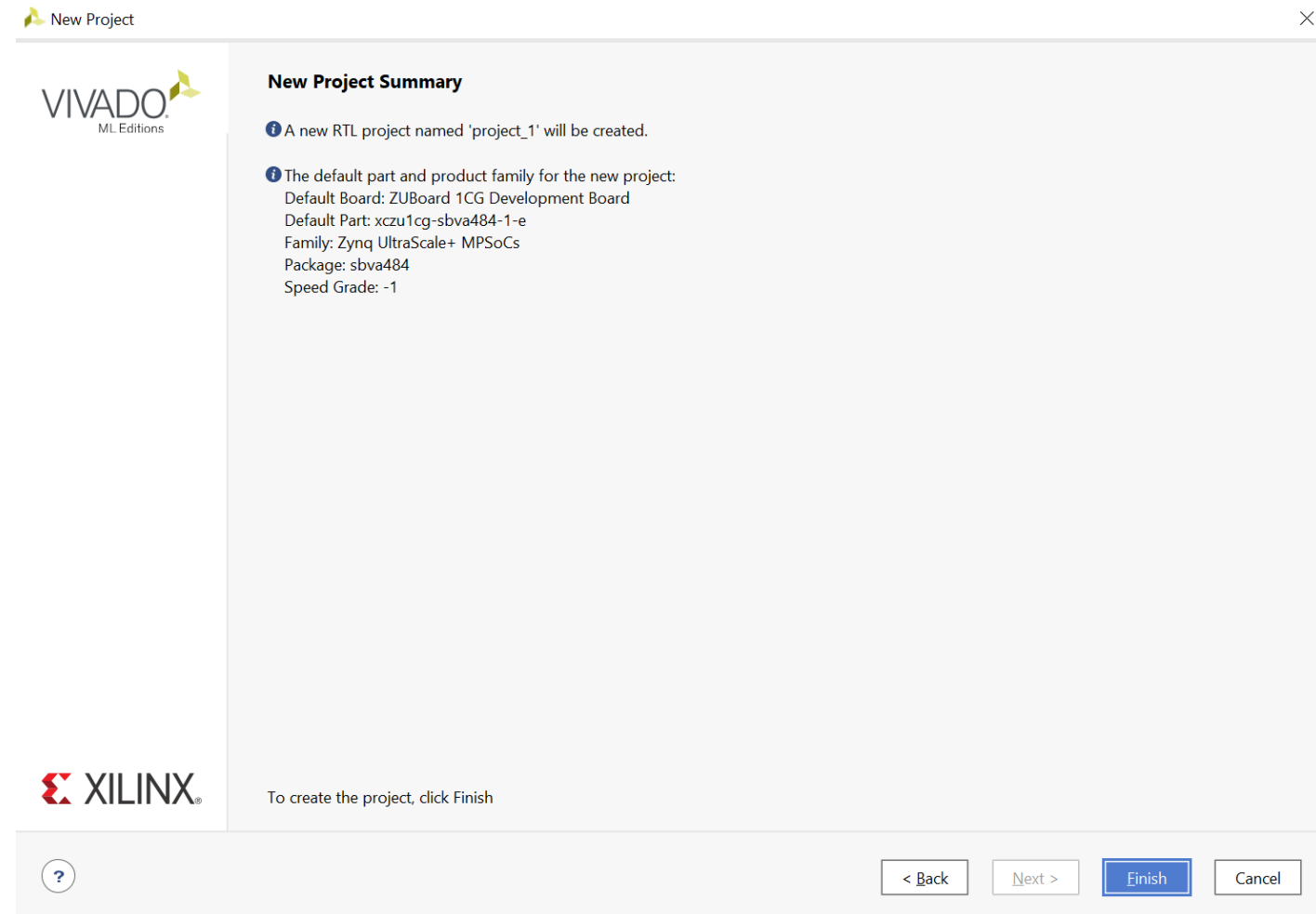
[Reset All Filters](#)

Vendor:
 Name:
 Board Rev:

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

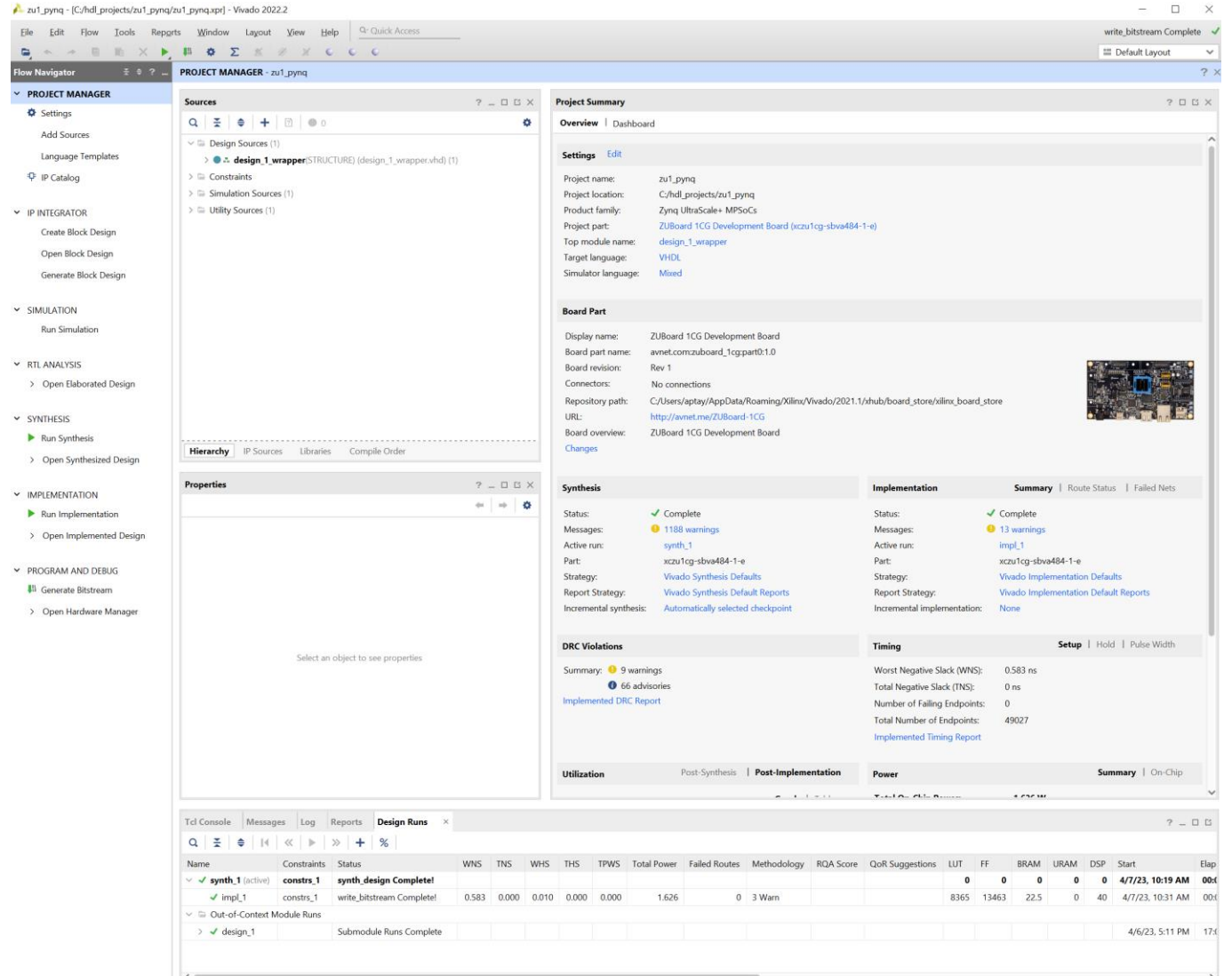
Lab : FFT Acceleration

Click Finish to create the project



Lab : FFT Acceleration

From the Project
Manager, select create
block diagram



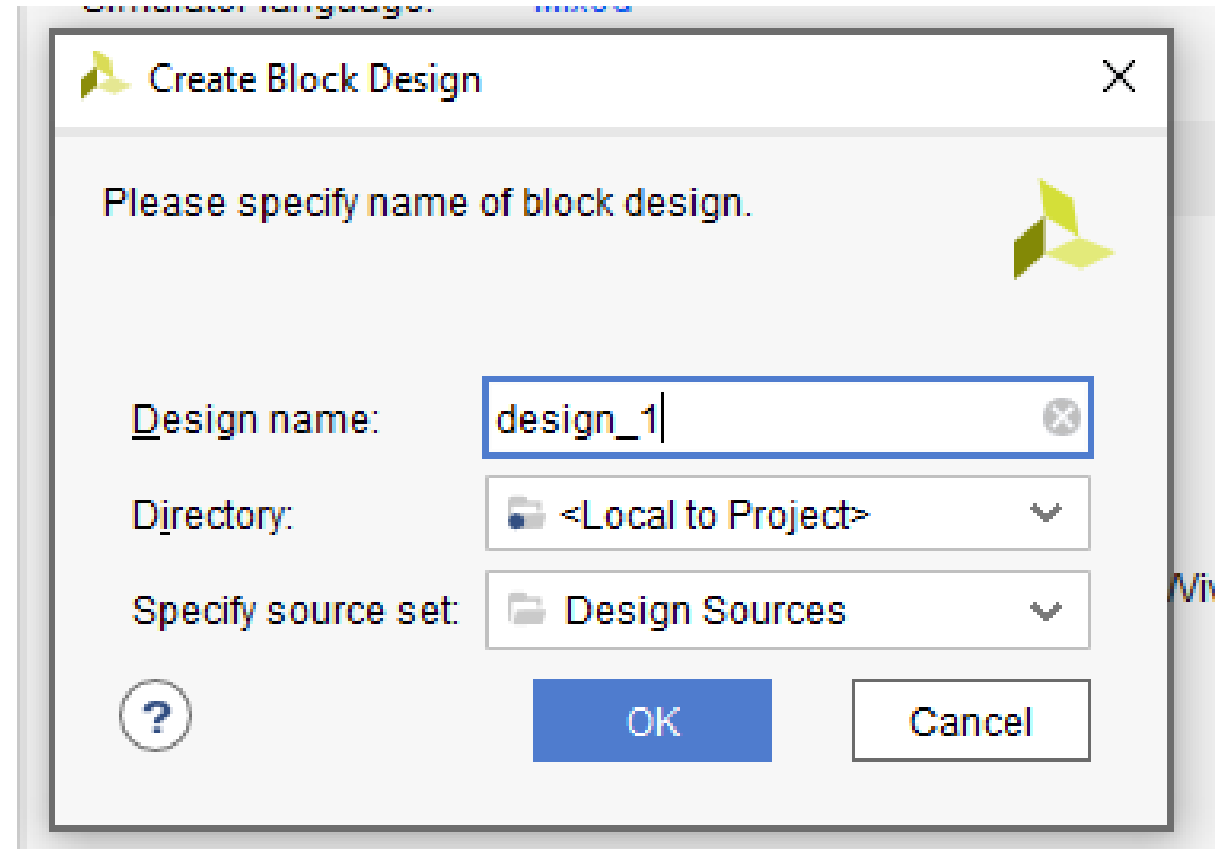
The screenshot displays the Vivado 2022.2 Project Manager interface for a project named 'zu1_pynq'. The left sidebar shows the 'PROJECT MANAGER' tree with options like 'Settings', 'Add Sources', 'Language Templates', 'IP Catalog', 'IP INTEGRATOR', 'SIMULATION', 'RTL ANALYSIS', 'SYNTHESIS', 'IMPLEMENTATION', and 'PROGRAM AND DEBUG'. The main area is divided into several panels:

- Sources:** Shows the project structure with 'design_1_wrapper' (STRUCTURE) and 'design_1_wrapper.vhd' (1).
- Project Summary:** Provides an overview of the project, including the project name, location, family (Zynq UltraScale+ MPSoCs), part (ZUBoard 1CG Development Board), and target language (VHDL).
- Board Part:** Details the board part, including the display name, board part name, revision, connectors, repository path, and board overview.
- Synthesis:** Shows the synthesis status as 'Complete' with 1188 warnings. The active run is 'synth_1' for the part 'xczu1cg-sbva484-1-e' using the 'Vivado Synthesis Defaults' strategy.
- Implementation:** Shows the implementation status as 'Complete' with 13 warnings. The active run is 'impl_1' for the same part using the 'Vivado Implementation Default Reports' strategy.
- DRC Violations:** Shows 9 warnings and 66 advisories. A link to the 'Implemented DRC Report' is provided.
- Timing:** Displays timing metrics such as Worst Negative Slack (WNS), Total Negative Slack (TNS), Number of Falling Endpoints, and Total Number of Endpoints.
- Utilization:** Shows utilization metrics for Post-Synthesis and Post-Implementation.
- Design Runs:** A table at the bottom summarizes the design runs, including the name, constraints, status, and completion time.

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

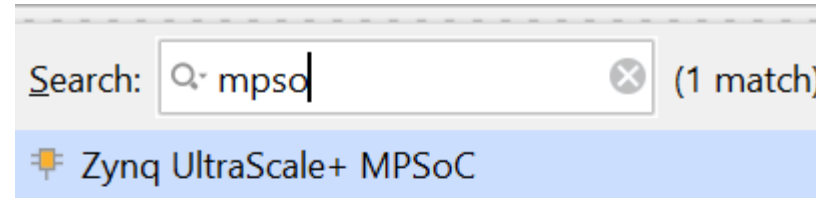
Lab : FFT Acceleration

Leave, defaults unchanged and click OK



Lab : FFT Acceleration

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 Acceleration

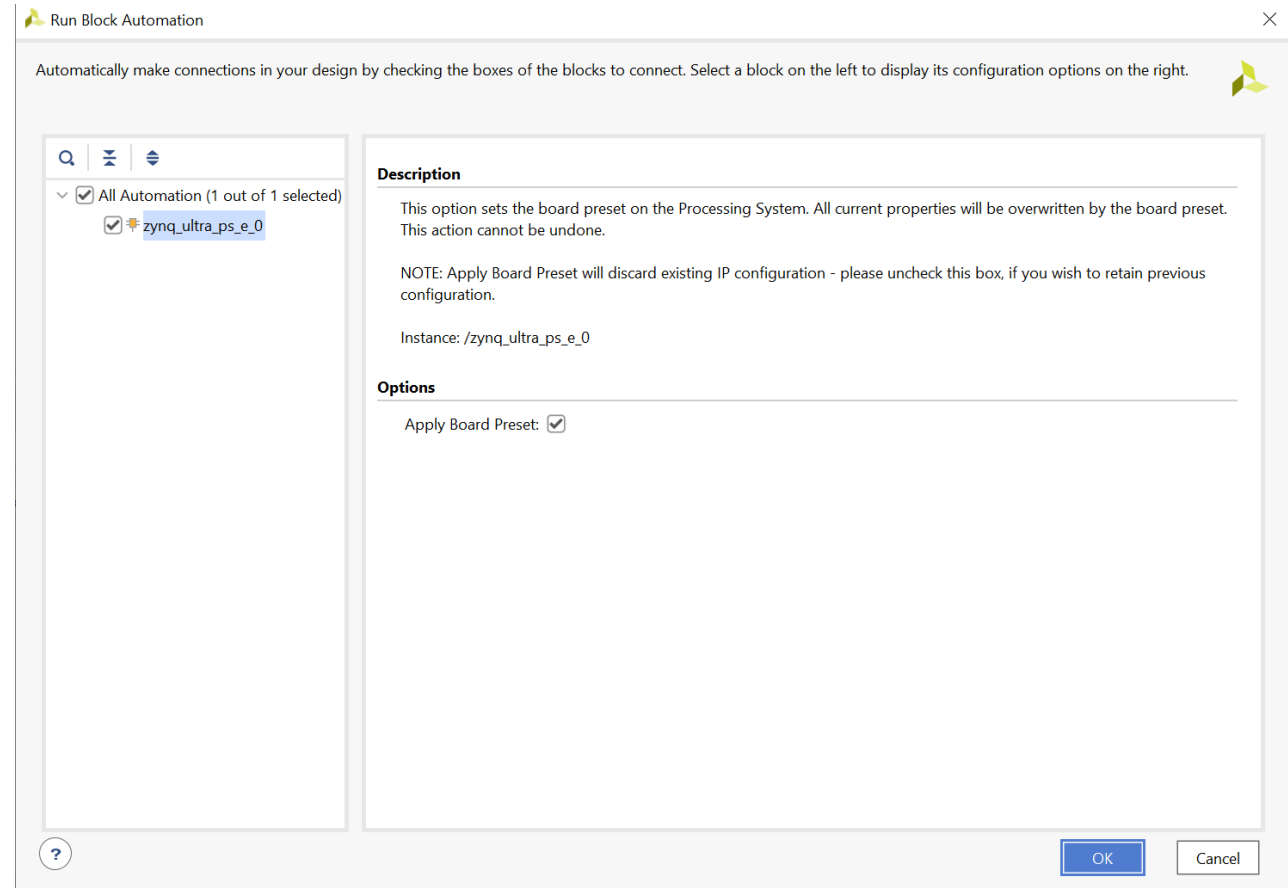
✦ Designer Assistance available. [Run Block Automation](#)

Run the block automation



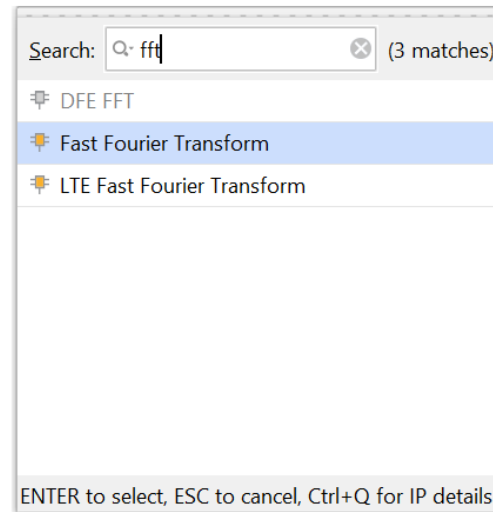
Lab : FFT Acceleration

Leave the settings as default and
click OK



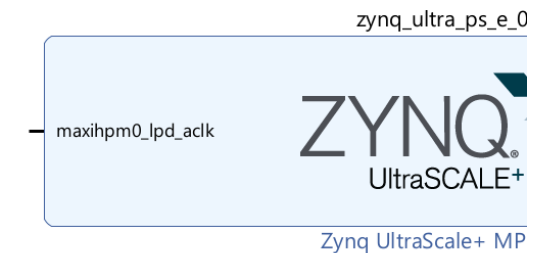
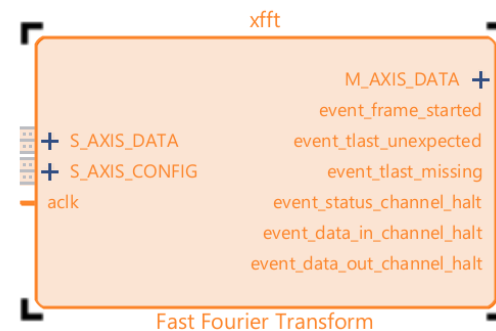
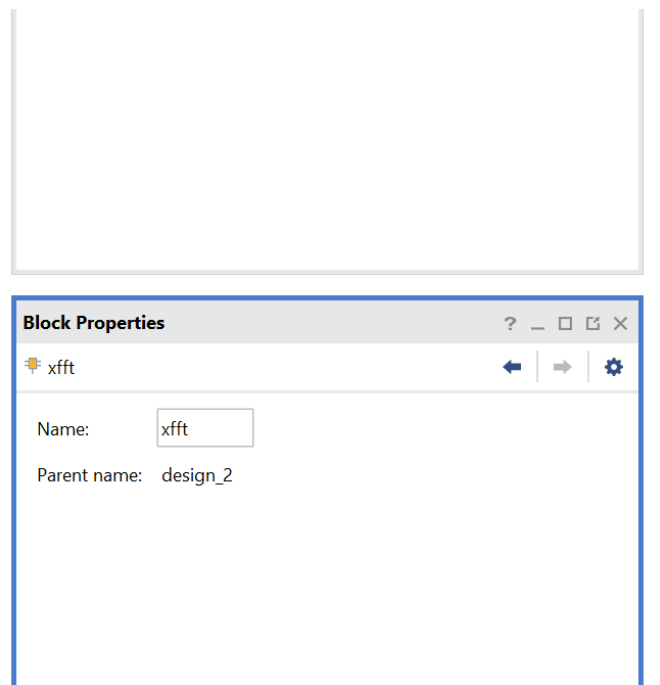
Lab : FFT Acceleration

Click on + and add in the
FFT



Lab : FFT Acceleration

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

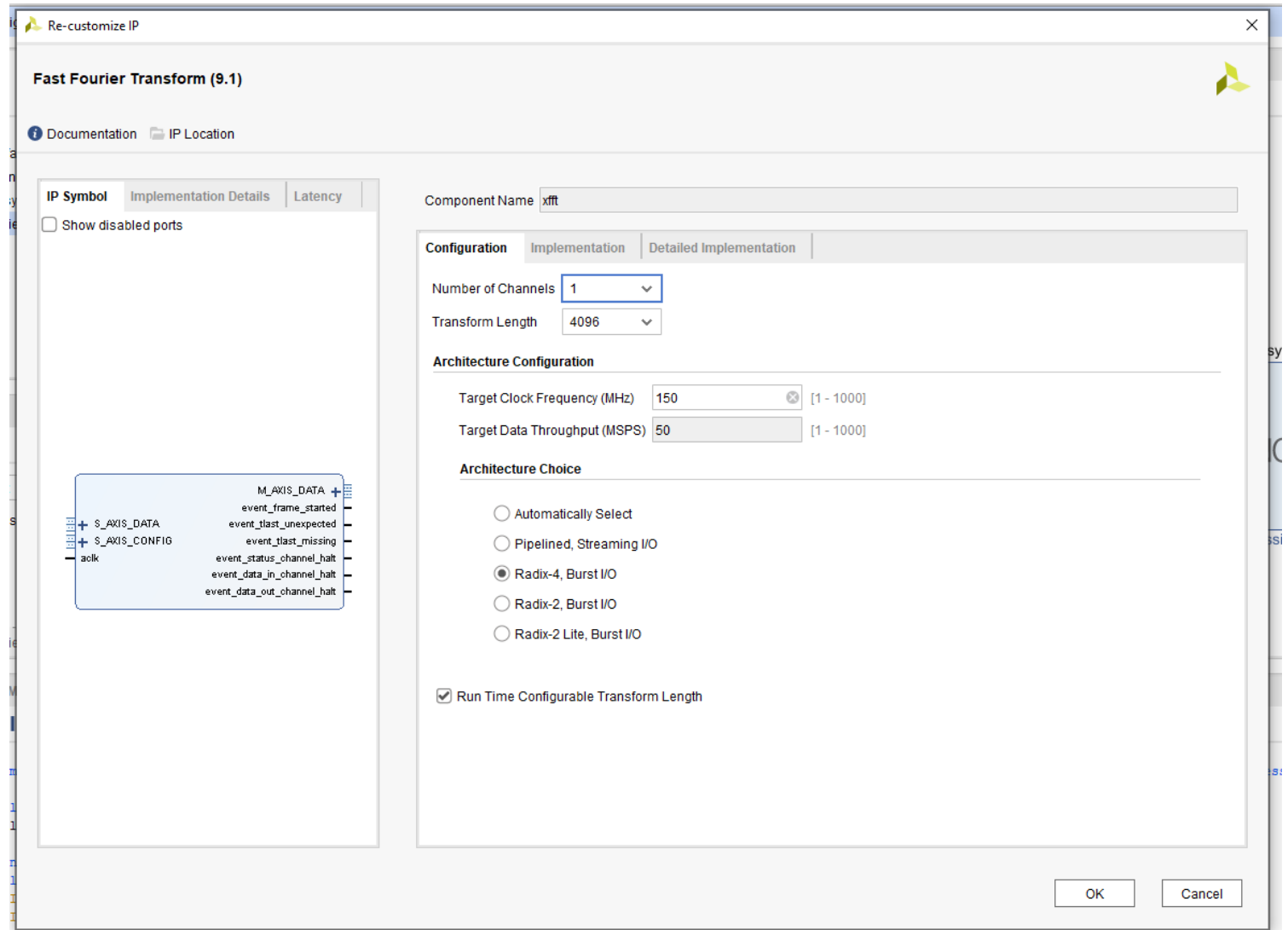


Lab : FFT Acceleration

On the configuration tab, select

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

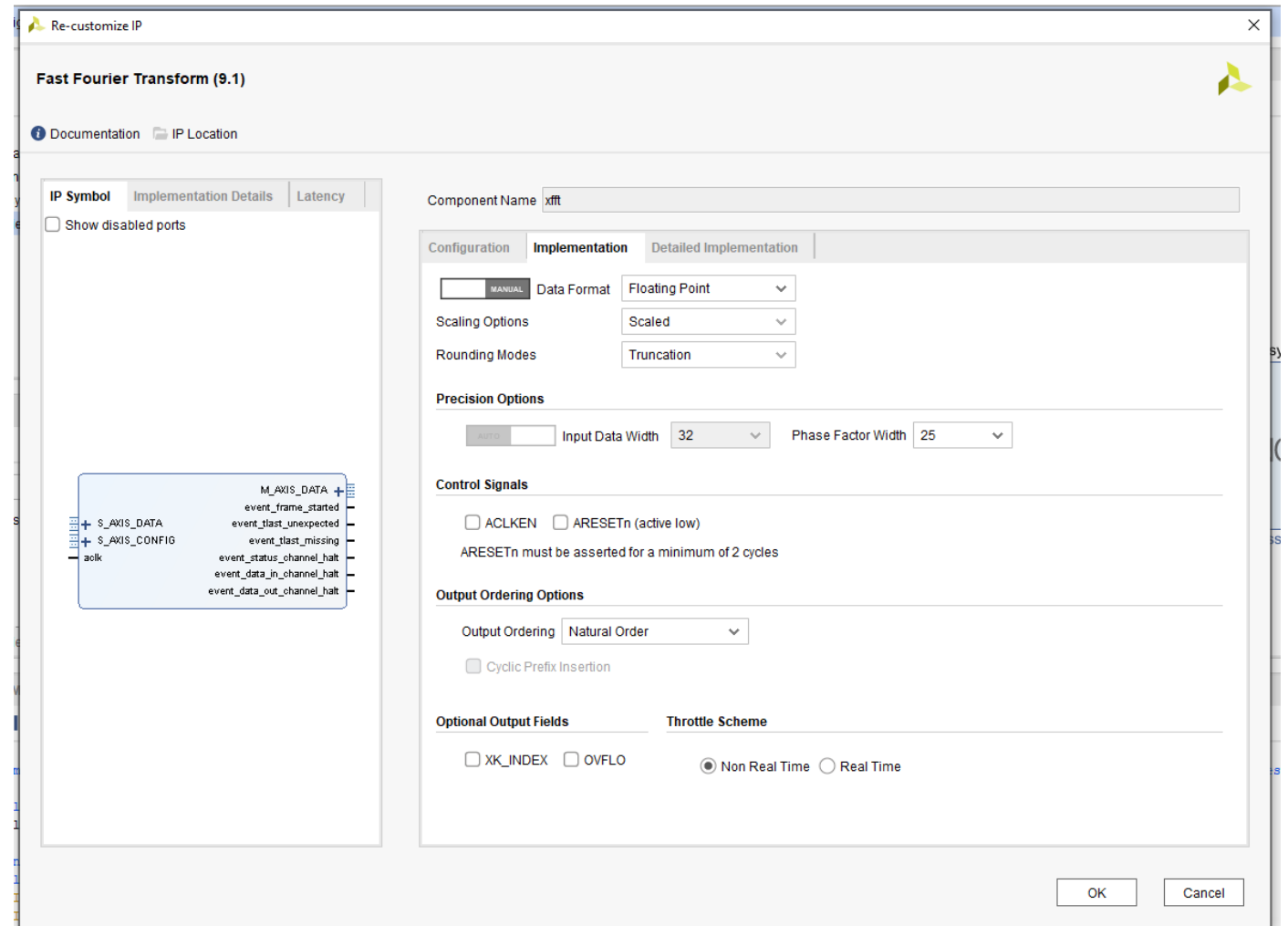
Configurable transform length



Lab : FFT Acceleration

On the implementation tab select

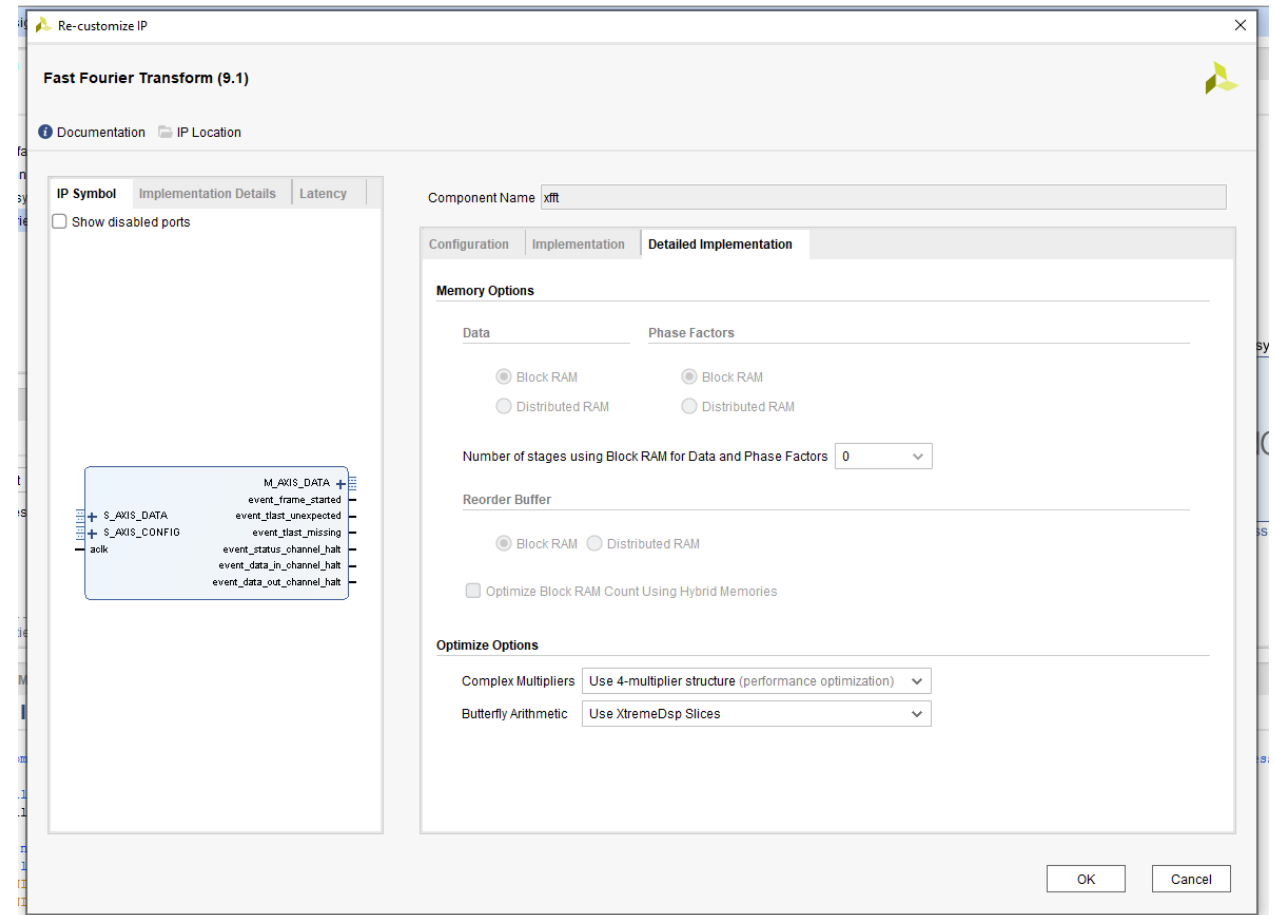
- Floating Point
- Phase Factor Width 25
- Output Ordering Natural
- Non-Real Time Throttle scheme



Lab : FFT Acceleration

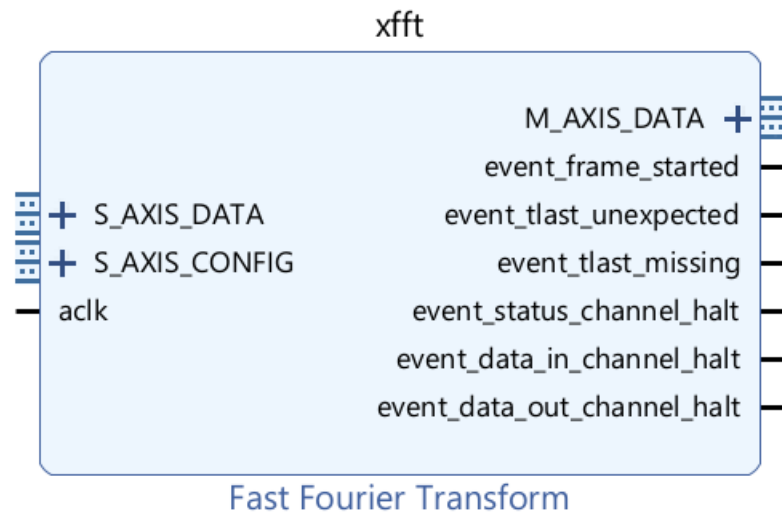
On the Detailed Implementation tab
select

- Use 4-Multiplier Structure
- Use XtremeDSP Slices



Lab : FFT Acceleration

Double click on the Processing System to reconfigure it



Lab : FFT Acceleration

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: Q-

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 Acceleration

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-

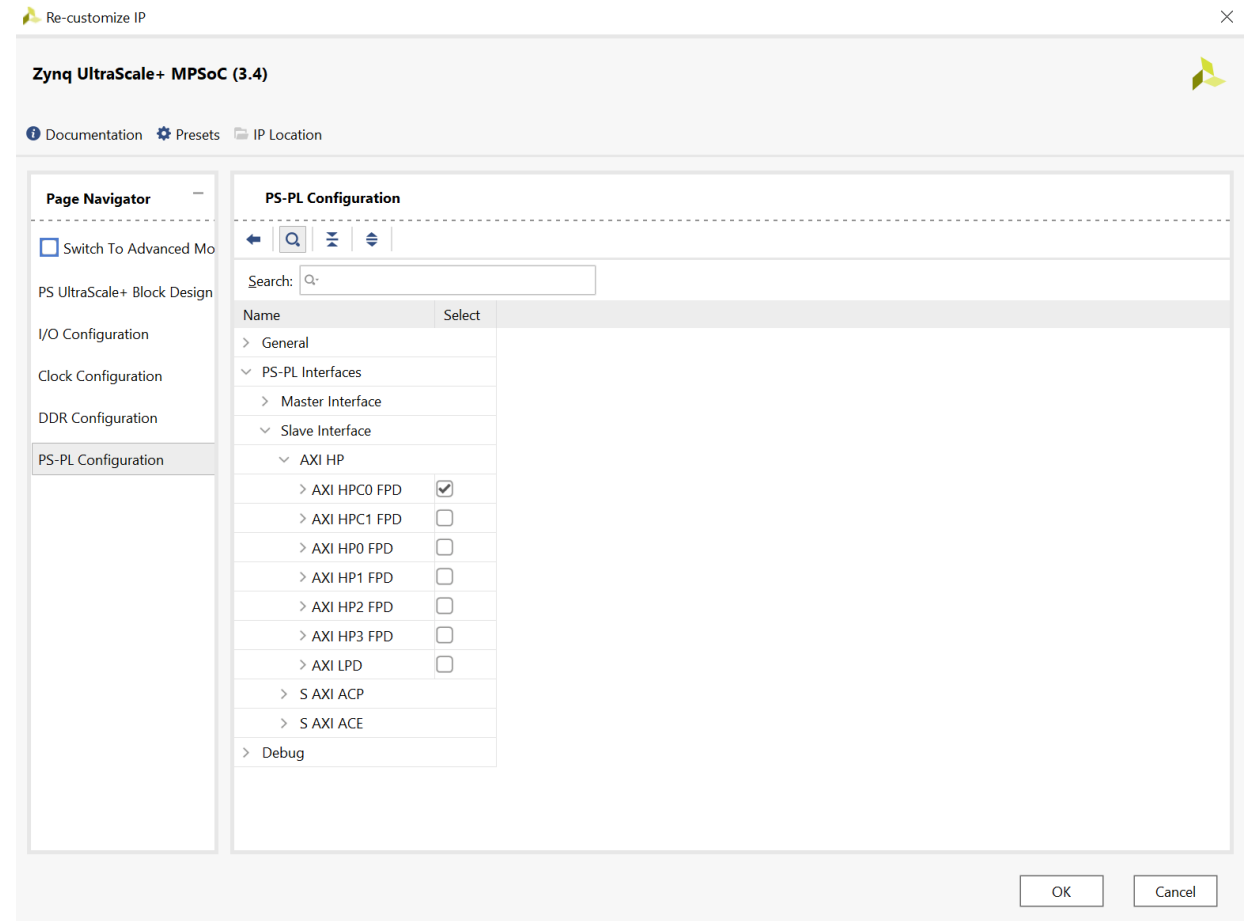
Name	Select
General	
Interrupts	
PL to PS	
IRQ0[0-7]	1
IRQ1[0-7]	0
APU Legacy Interr...	<input type="checkbox"/>
RPU Legacy Interr...	<input type="checkbox"/>
PS to PL	
Fabric Reset Enable	<input checked="" type="checkbox"/>
Address Fragmentation	
Others	
PS-PL Interfaces	
Master Interface	
Slave Interface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

OK Cancel

Lab : FFT Acceleration

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

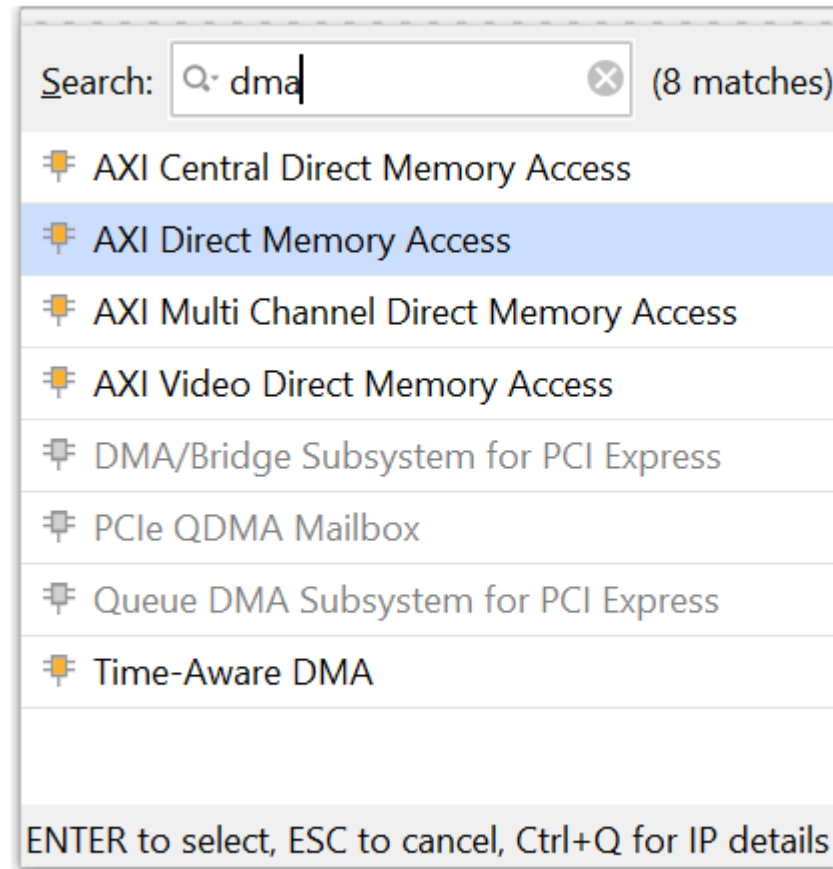
Enable AXI HPC0 FPD Interface



Lab : FFT Acceleration

Click + and select AXI

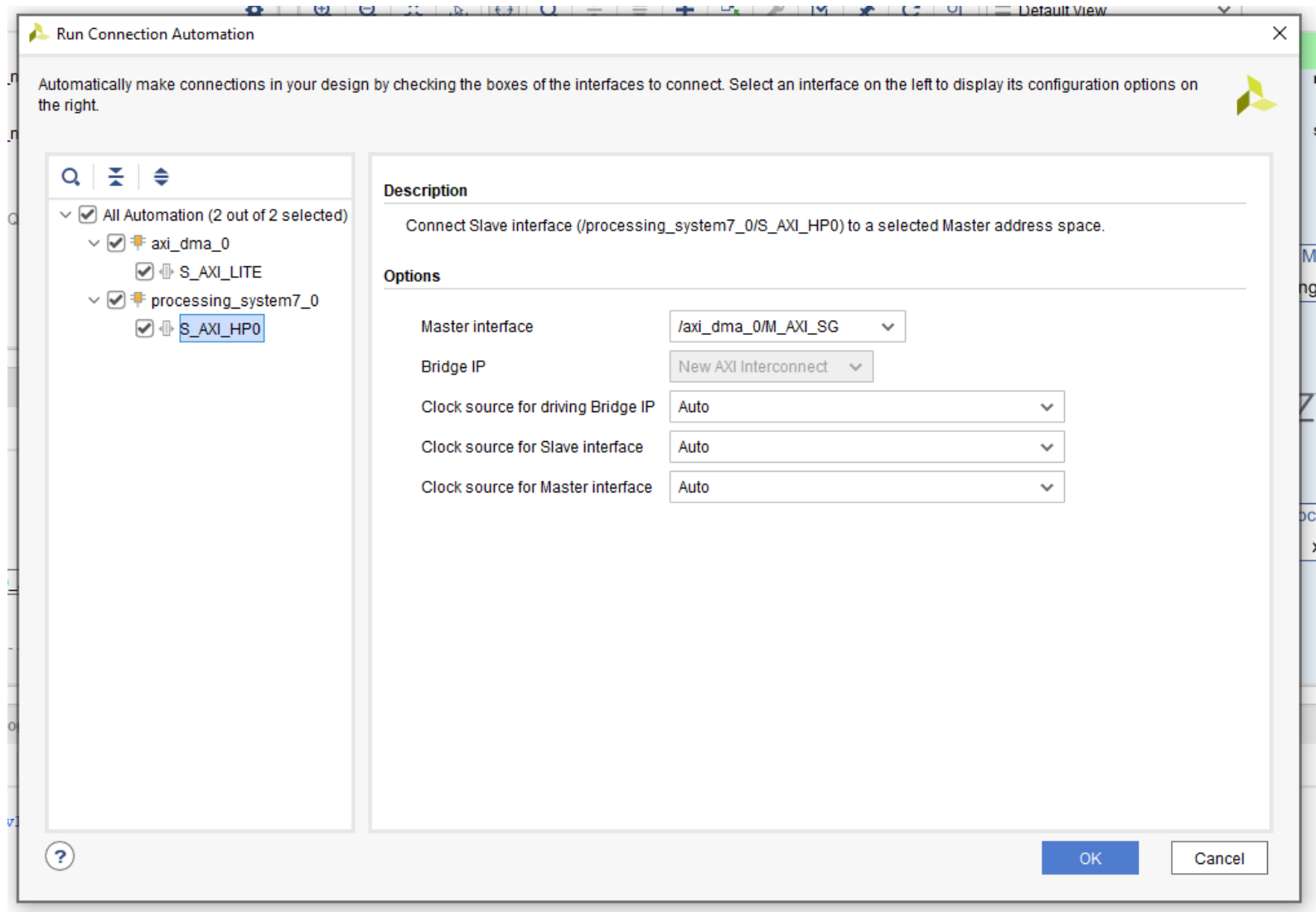
Direct Memory Access



Lab : FFT Acceleration

Run the connection
automation

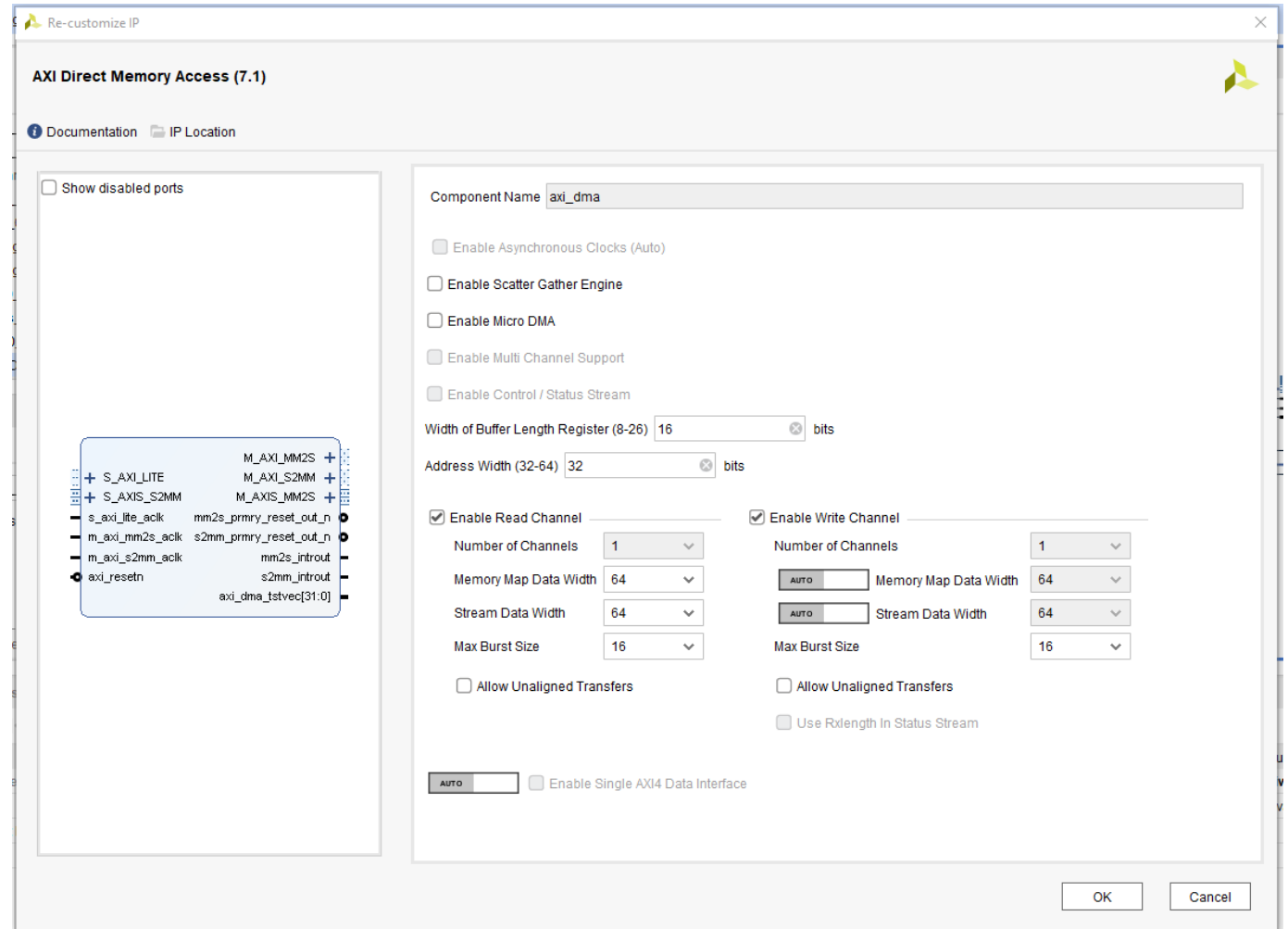
Leave the defaults as
standard and click OK



Lab : FFT Acceleration

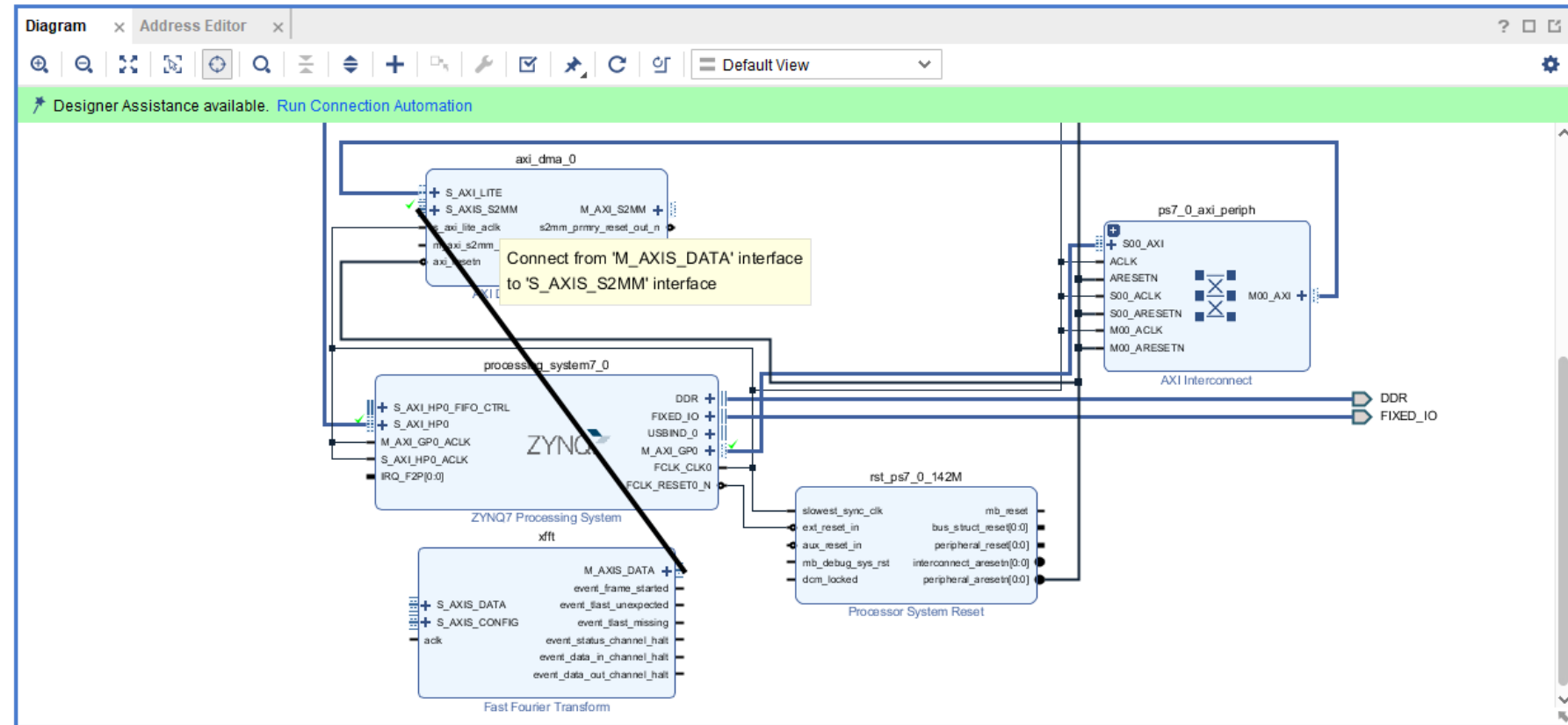
Select the DMA, double click on it
and configure it

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



Lab : FFT Acceleration

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

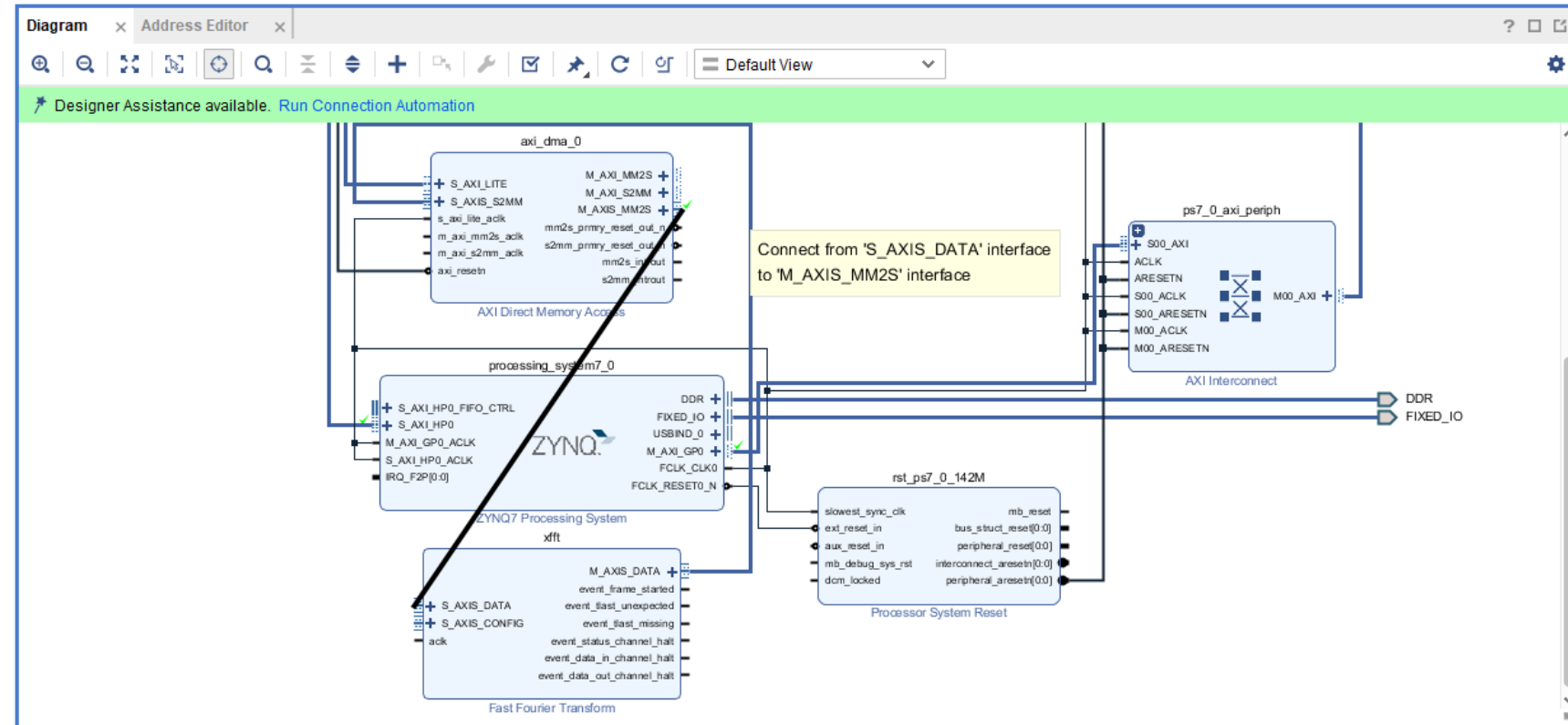


Lab : FFT Acceleration

Connect the DMA M

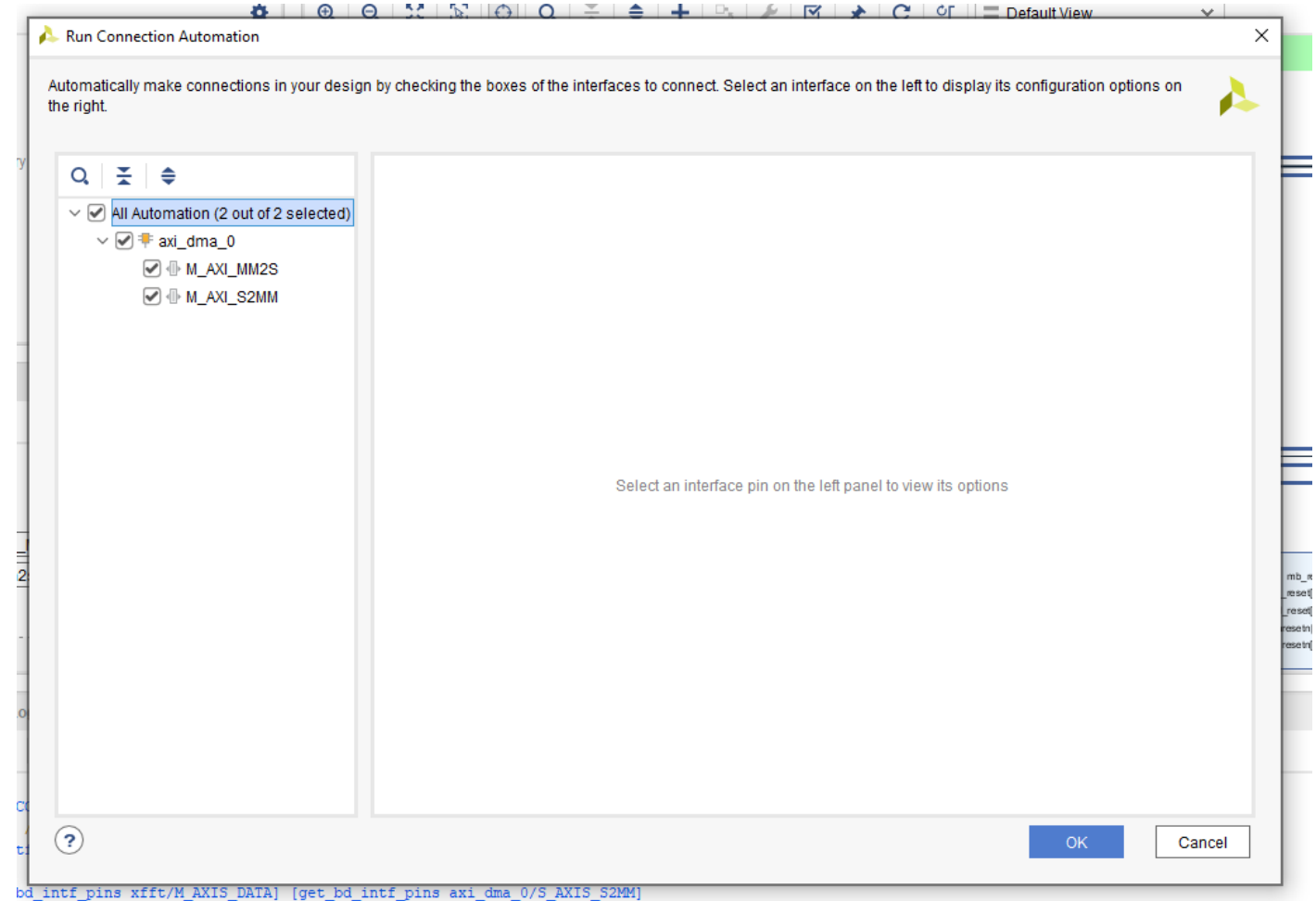
AXIS MM2S to the xFFT

S AXIS Data



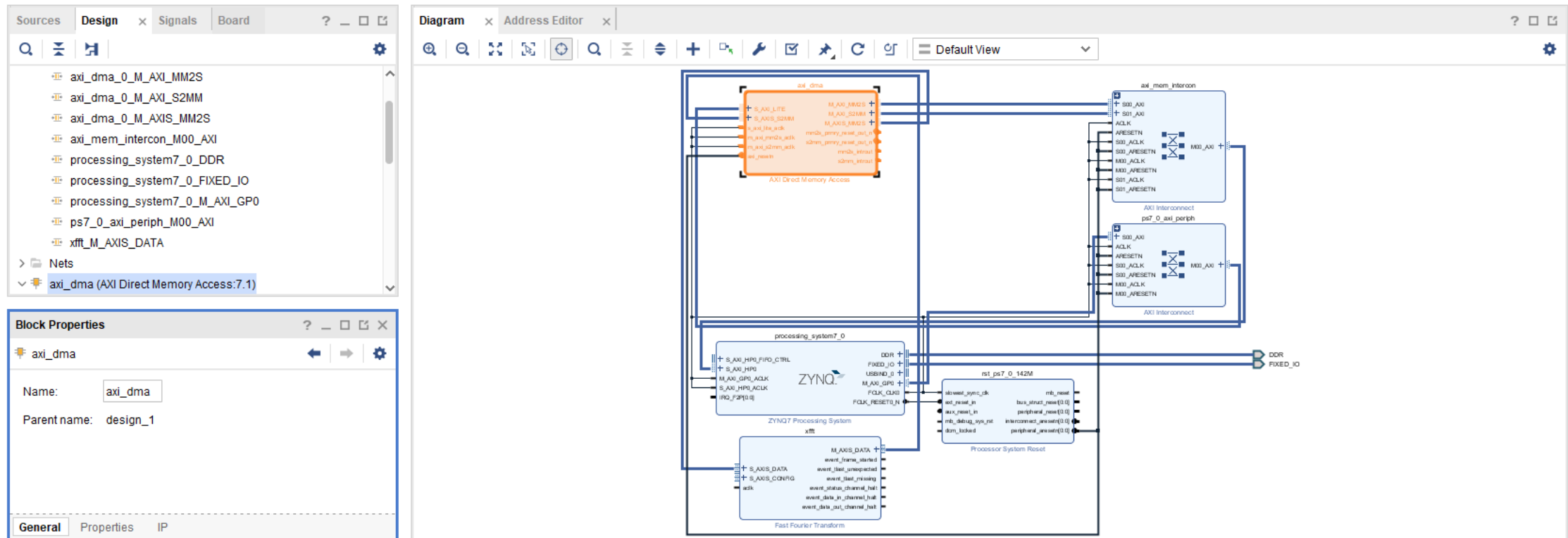
Lab : FFT Acceleration

Run the connection automation

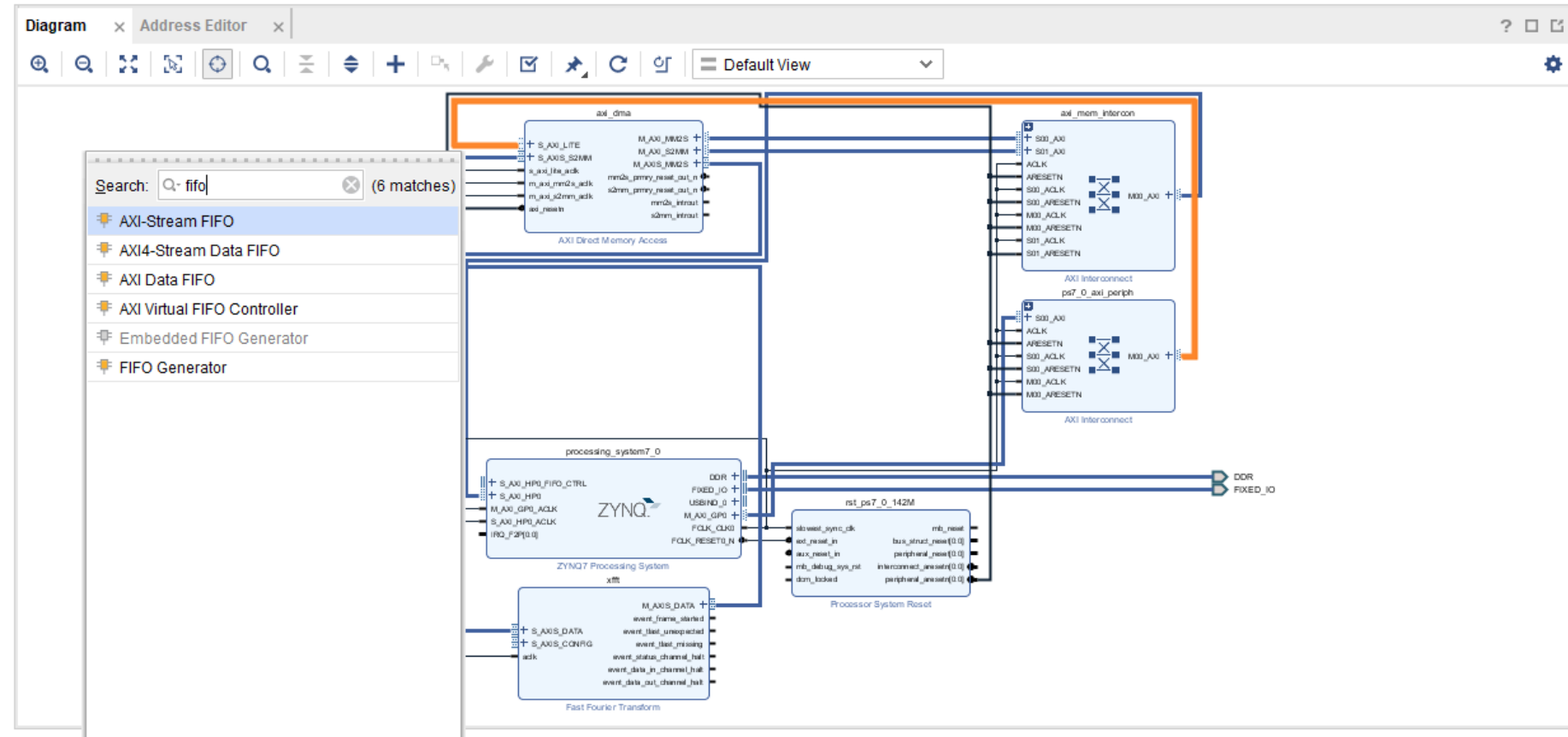


Lab : FFT Acceleration

The diagram should look like below



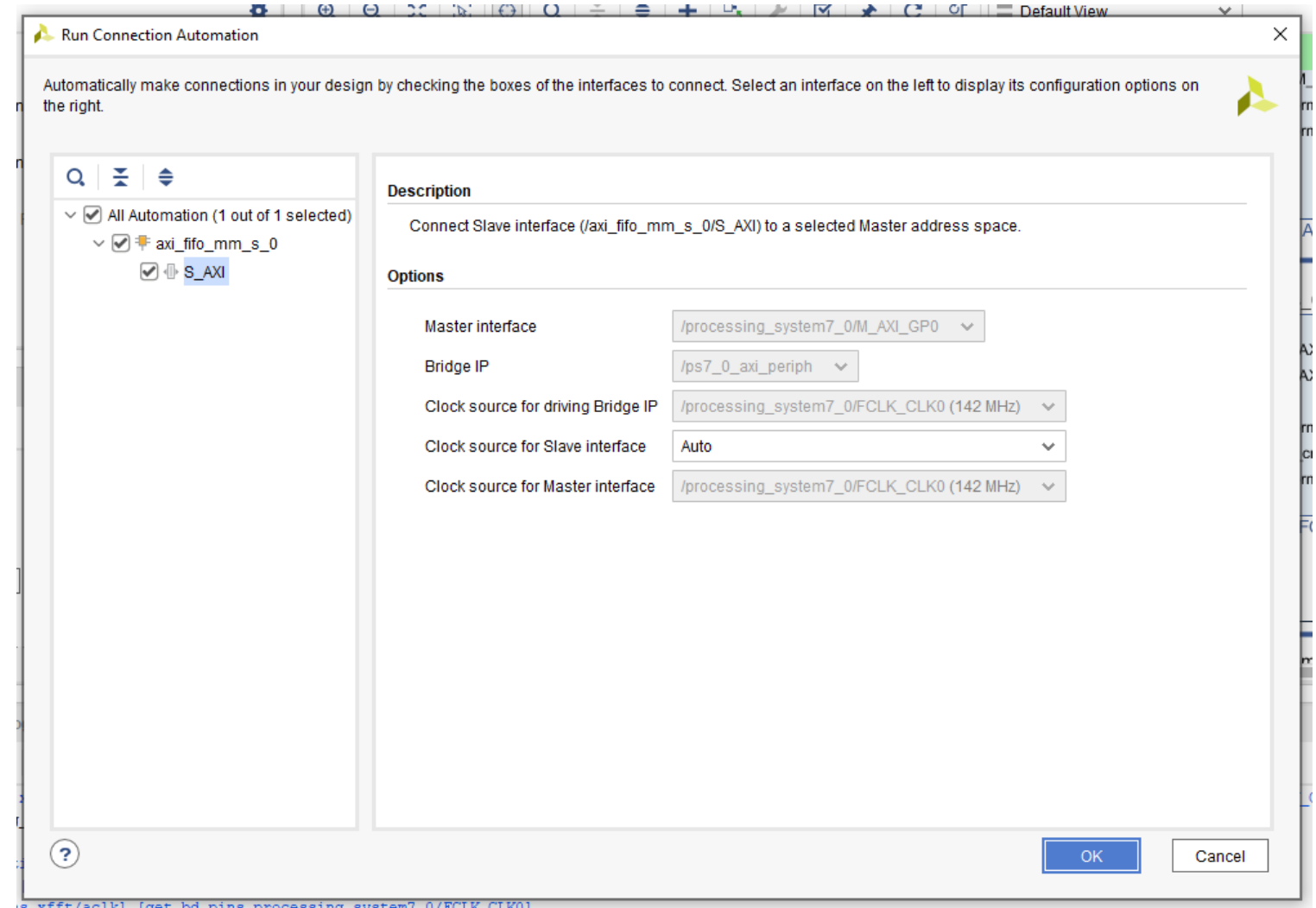
Click on + and add
in an AXI-Stream
FIFO



Lab : FFT Acceleration

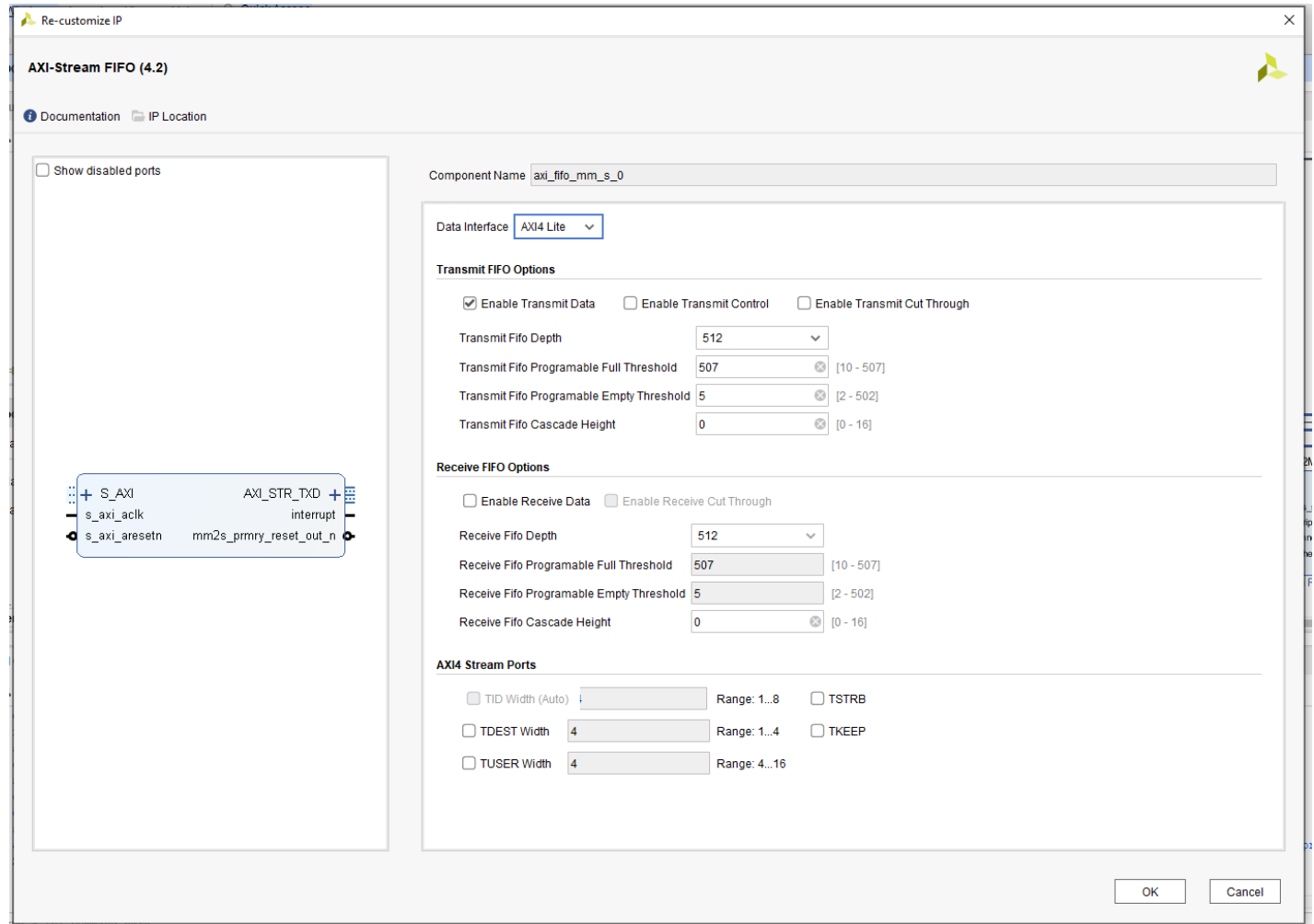
Run the connection

automation and click on OK



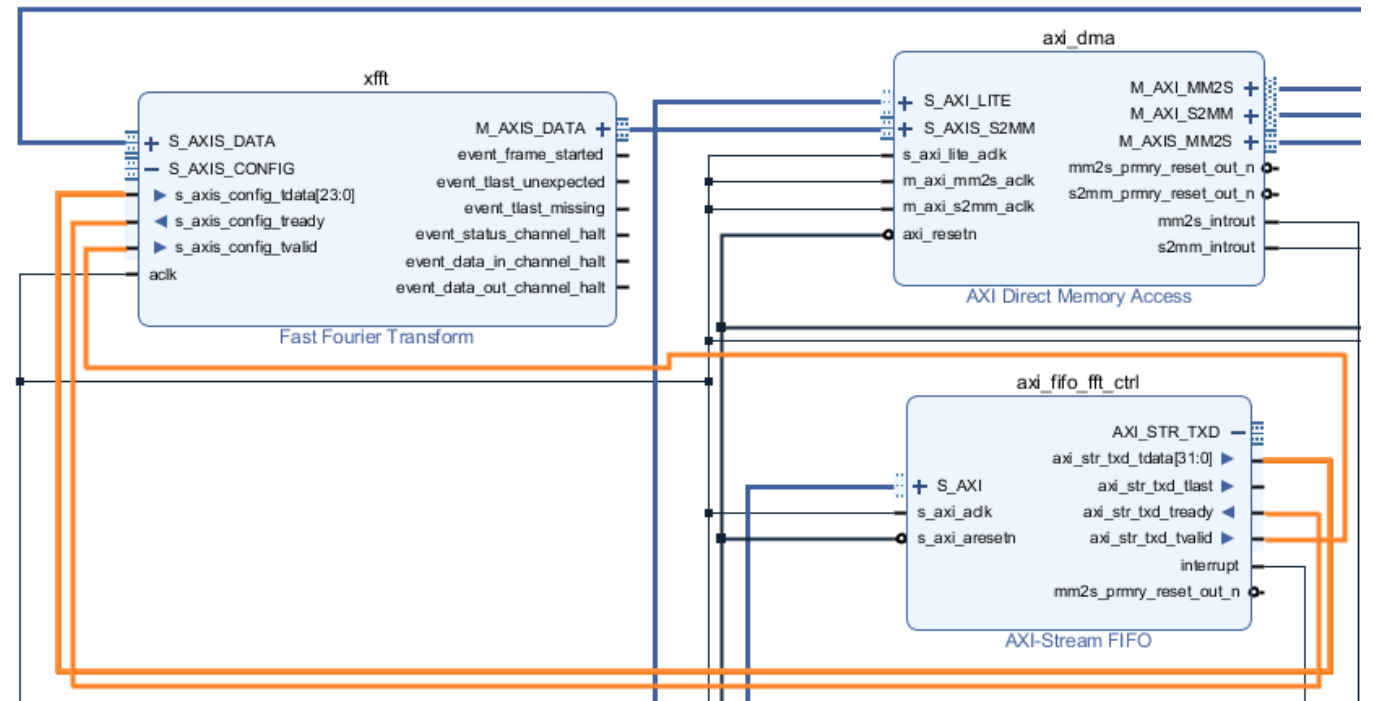
Lab : FFT Acceleration

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 Acceleration

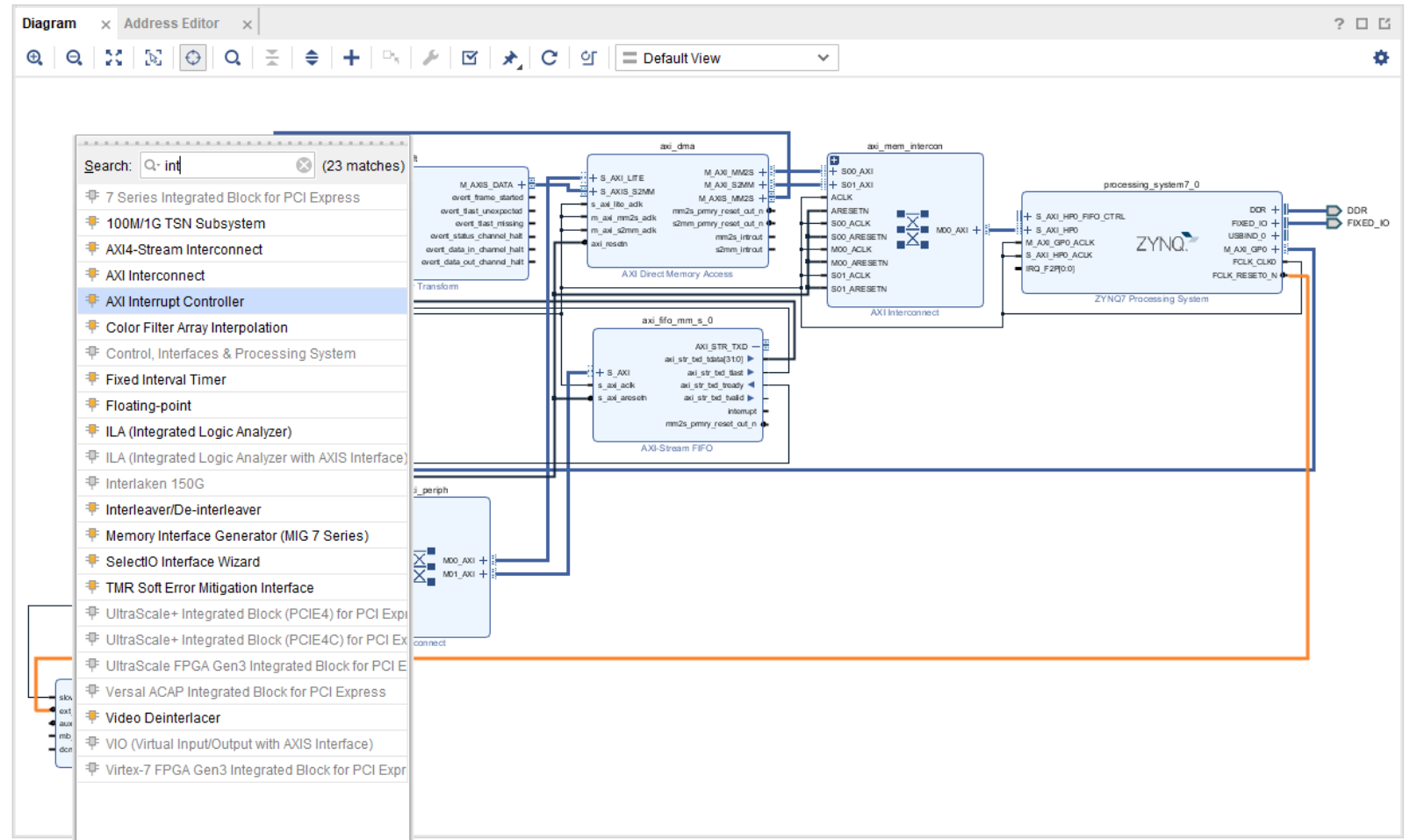
Connect the AXI STR TXD tdata, tlast and tvalid signals to the xFFT S AXIS config



Lab : FFT Acceleration

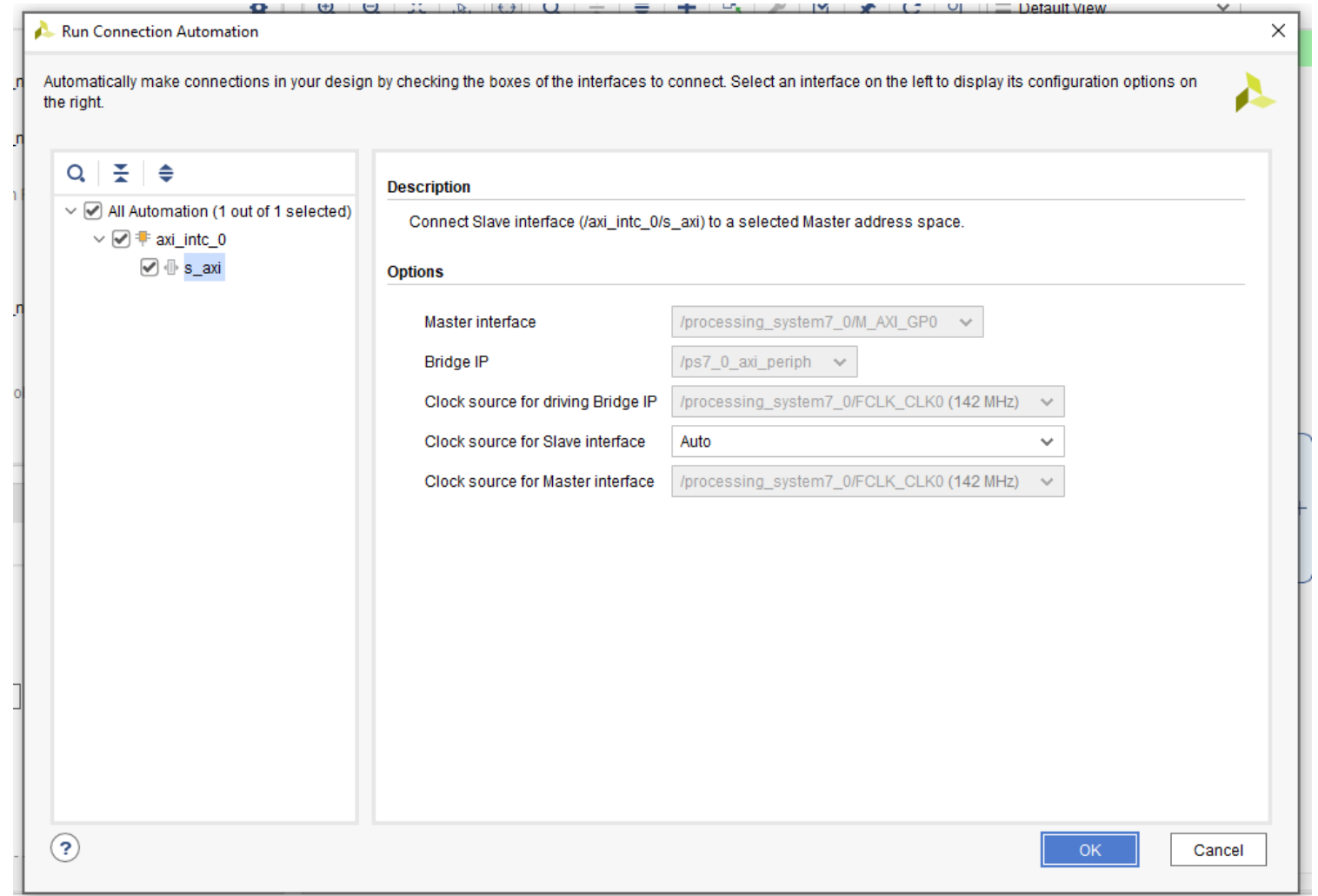
Click on + and add in a AXI

Interrupt Controller



Lab : FFT Acceleration

Run the connection
automation



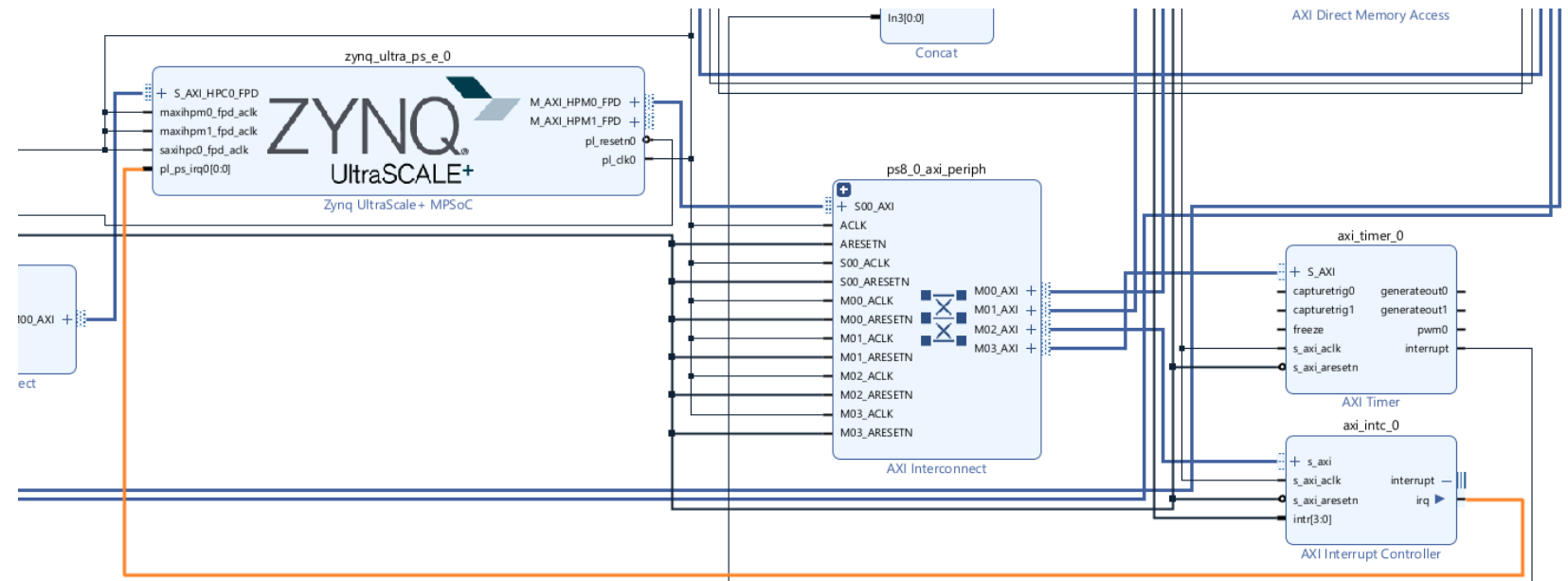
Lab : FFT Acceleration

Connect the IRQ output

from the AXI Interrupt

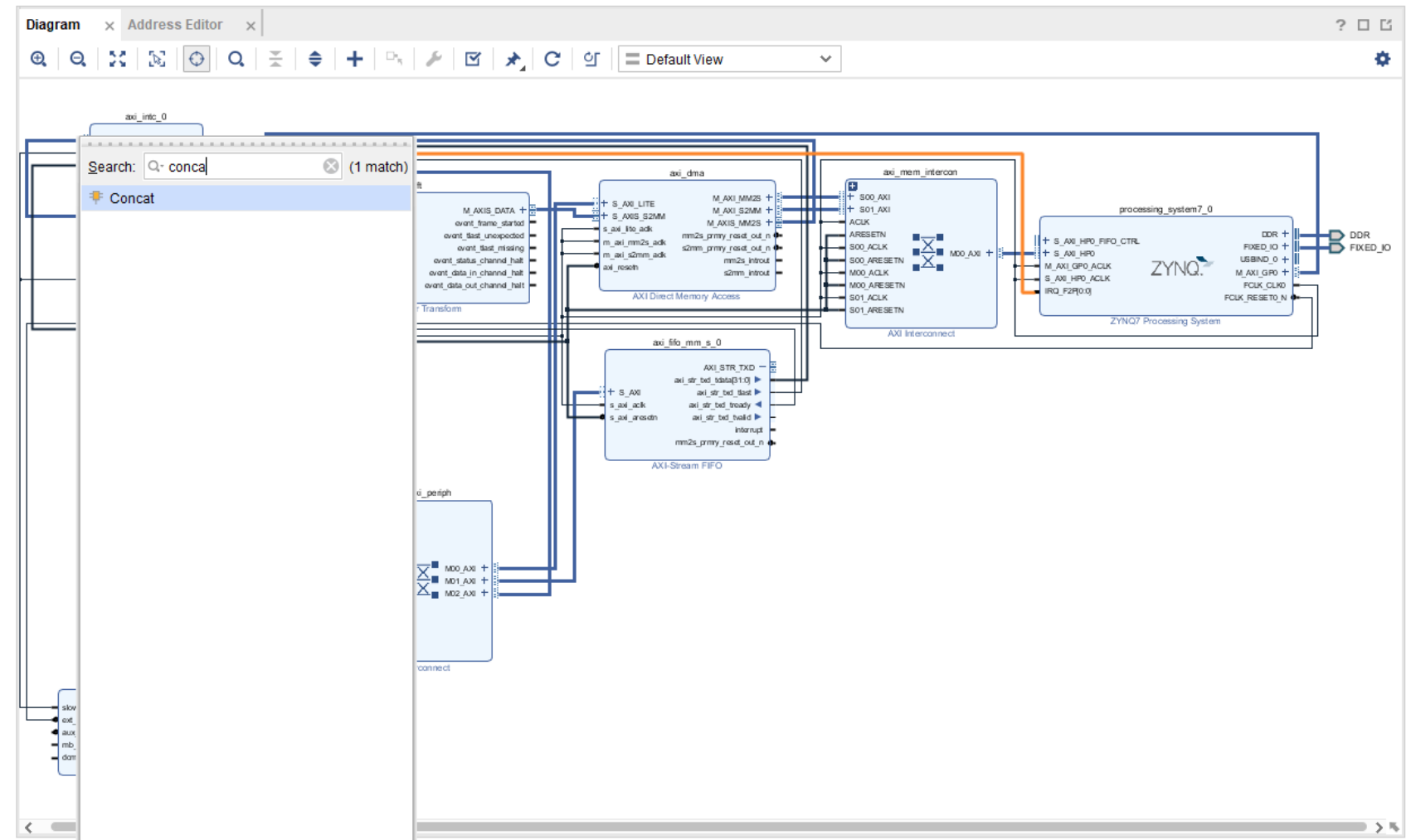
Controller to the PL PS

IRQ port on the Zynq



Lab : FFT Acceleration

Click on + and add in a
concat block



Lab :

Go to PYM

Re-customize IP

Concat (2.1)

[Documentation](#) [IP Location](#)

☐ Show disabled ports

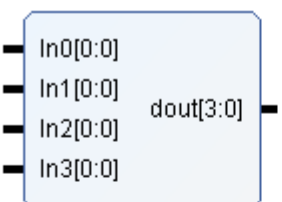
Component Name

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]

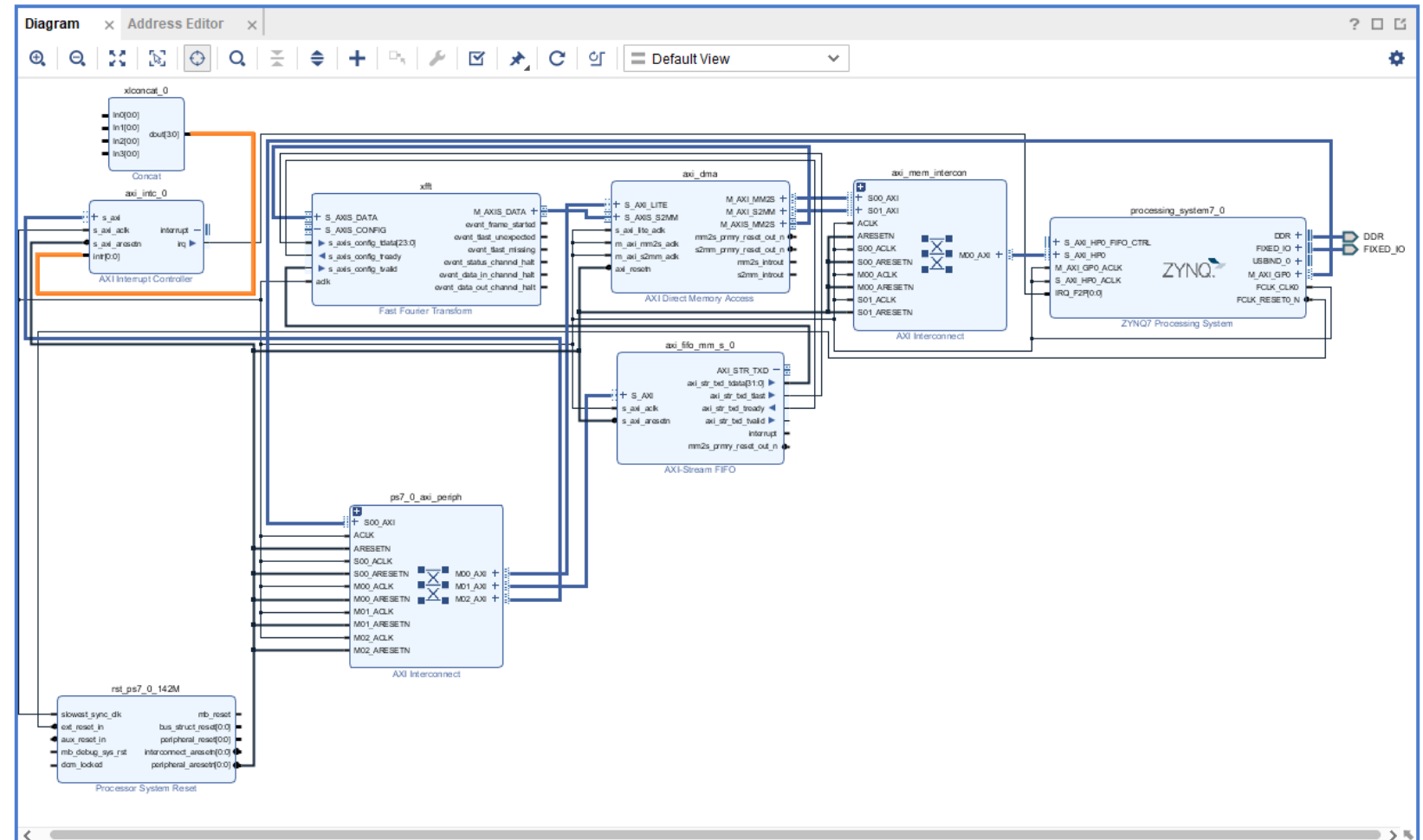
Dout Width (Auto)

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.



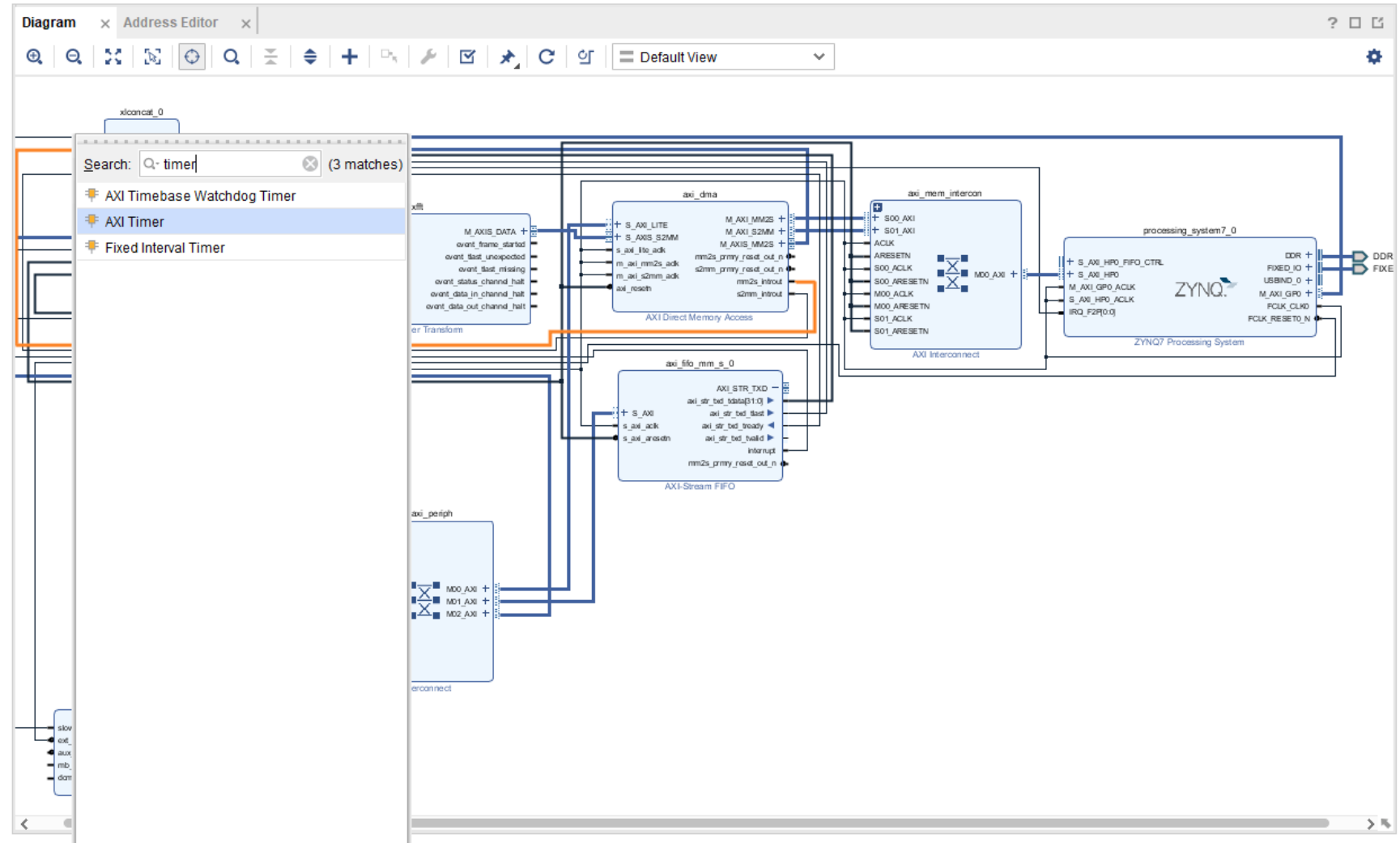
Lab : FFT Acceleration

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



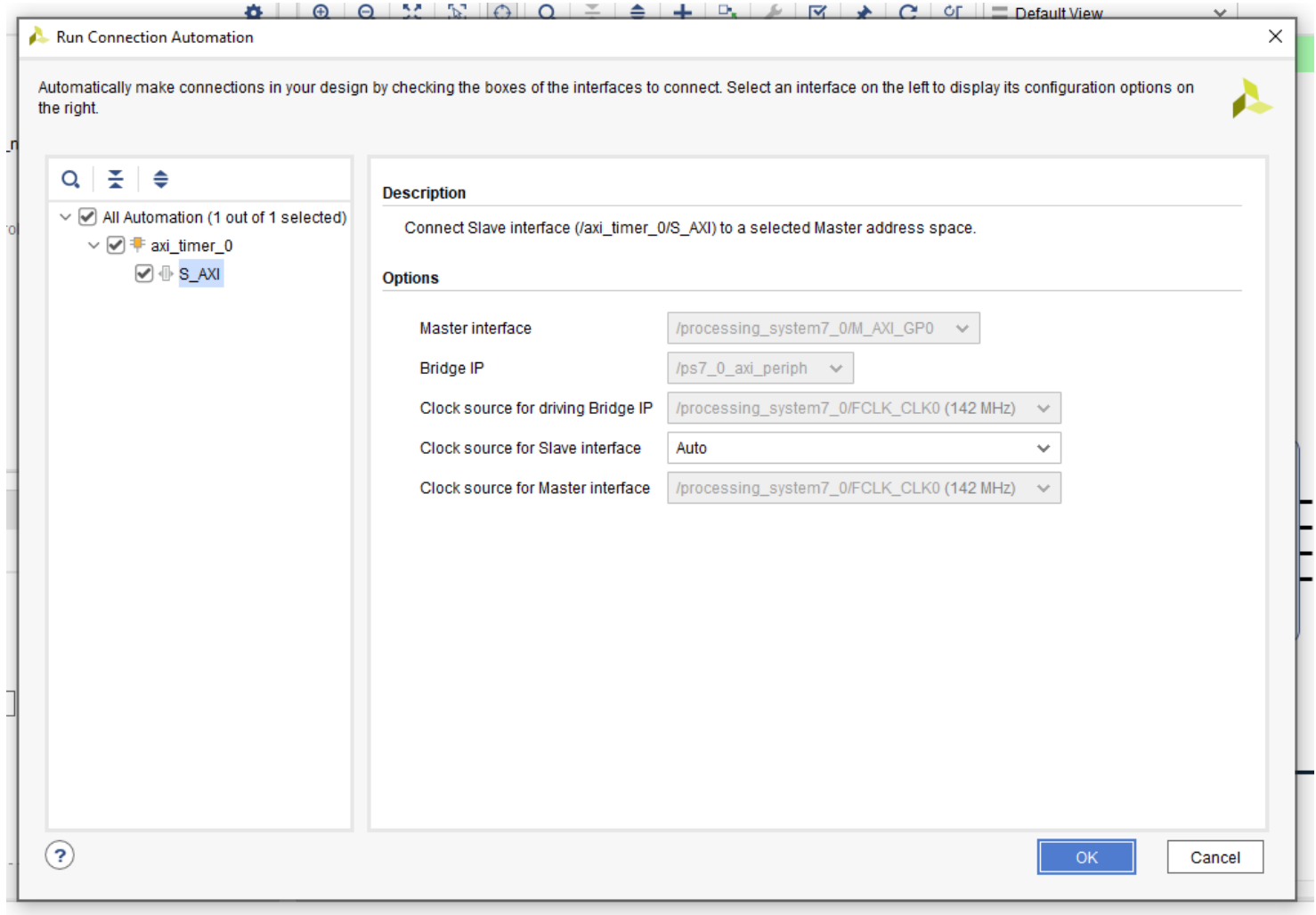
Lab : FFT Acceleration

Click on the + symbol
and add in a AXI Timer



Lab : FFT Acceleration

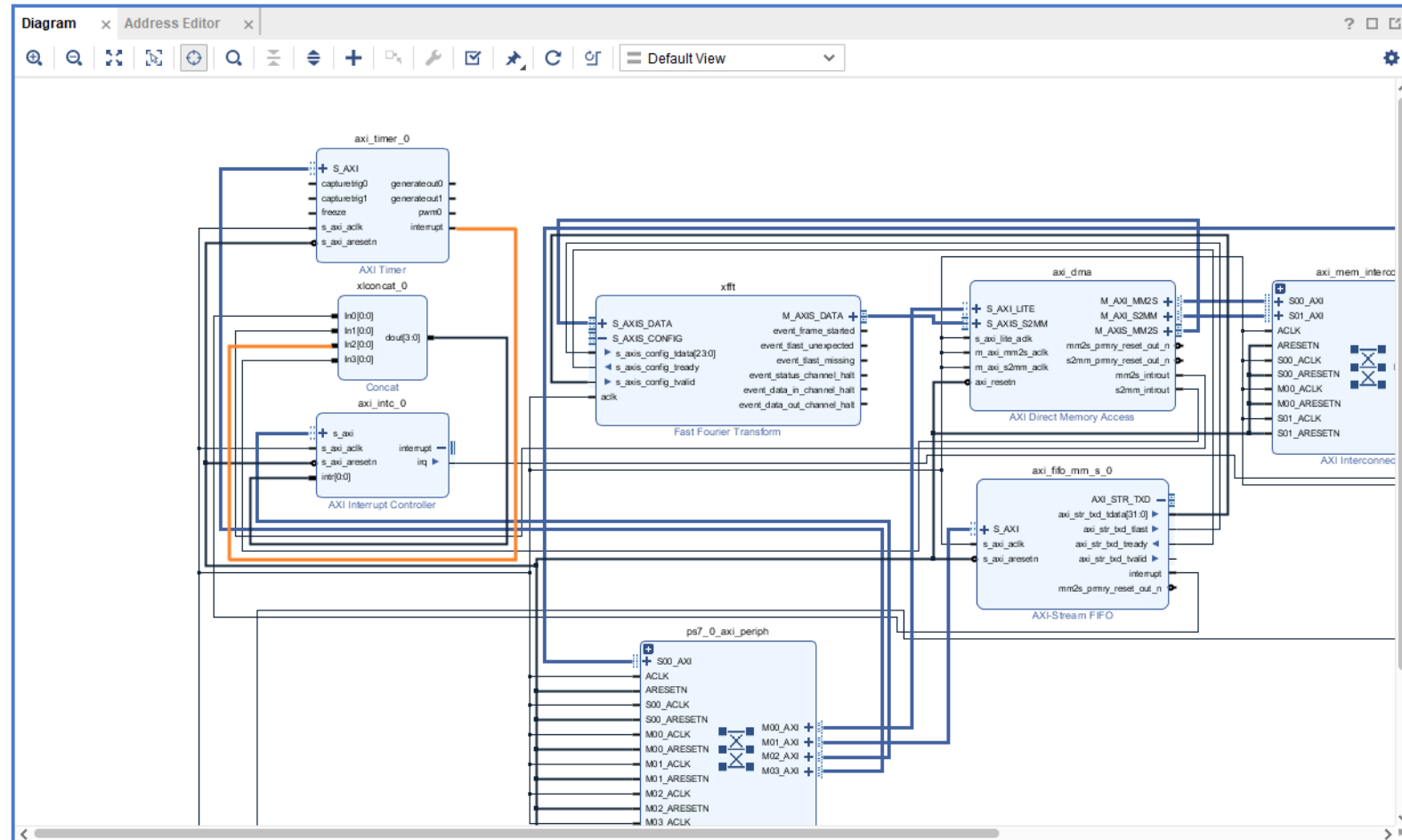
Run the connection automation



Lab : FFT Acceleration

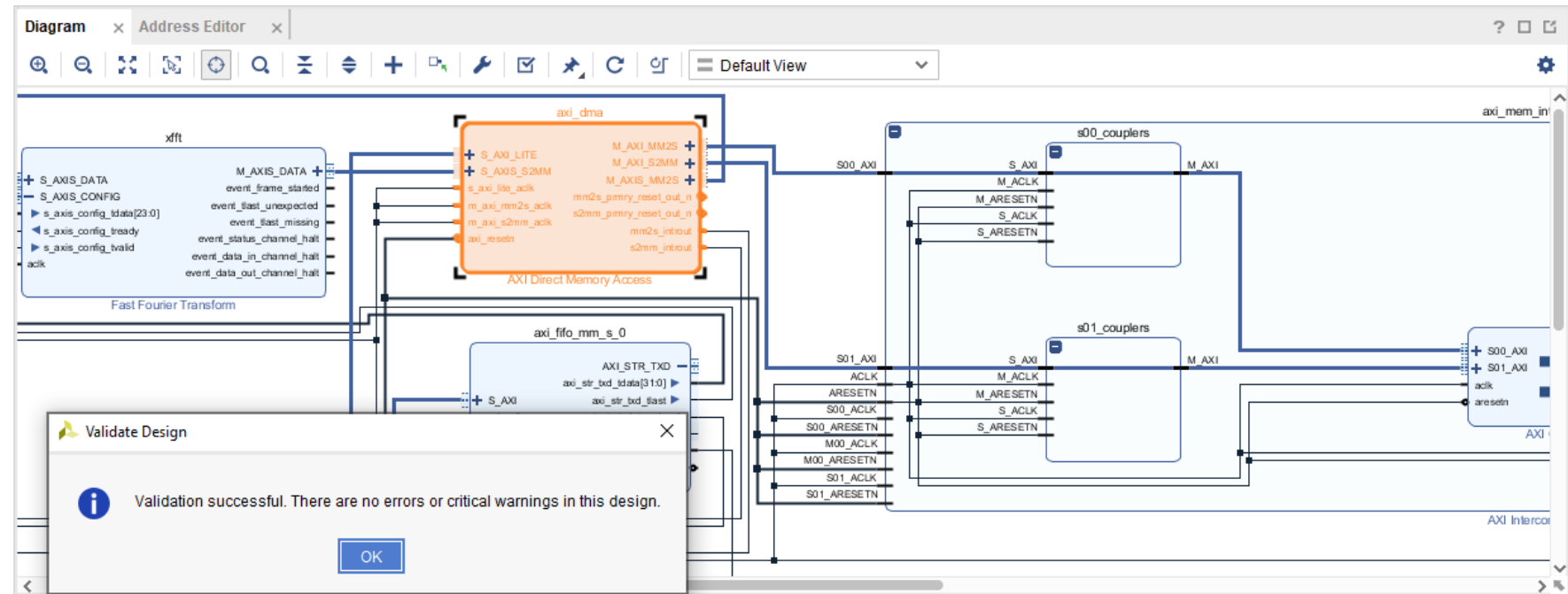
Connect the interrupts to the
concat block

- AXI Timer
- AXI DMA MM2S_INTOUT
- AXI DMA S2MM_INTOUT
- AXI Stream FIFO



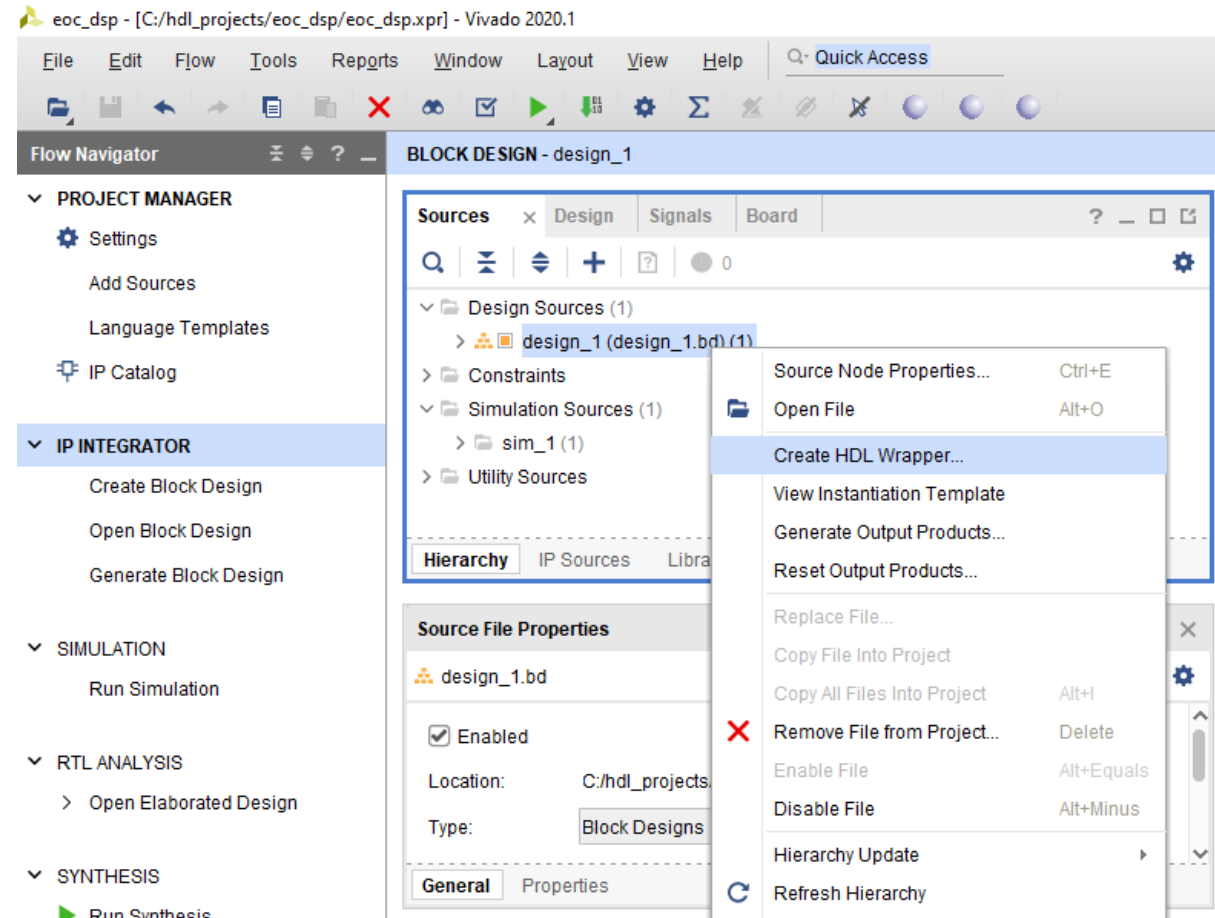
Lab : FFT Acceleration

Validate the design
there should be no
error or critical
warnings



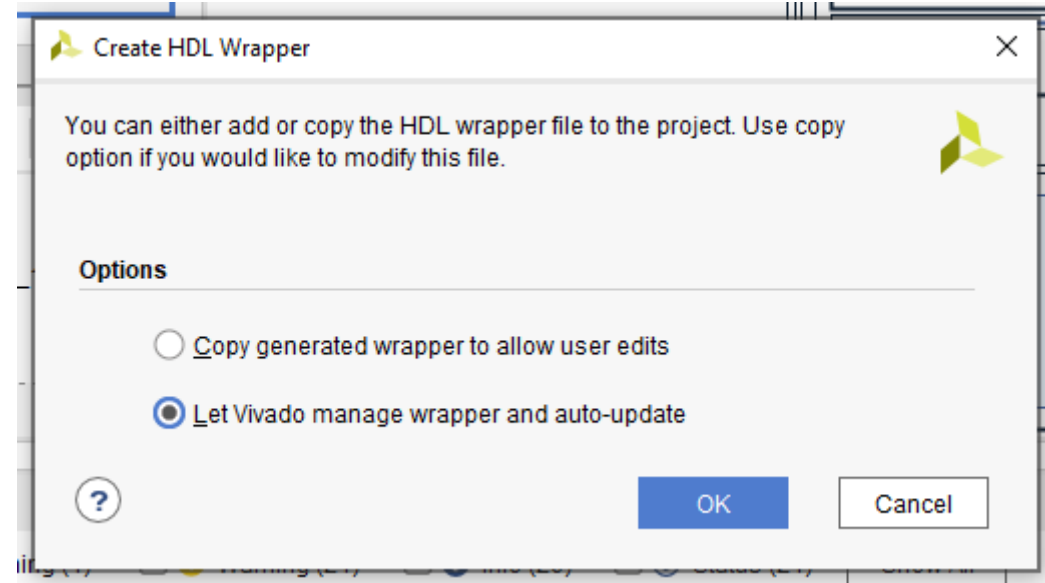
Lab : FFT Acceleration

Right click on the design and
select Create HDL Wrapper



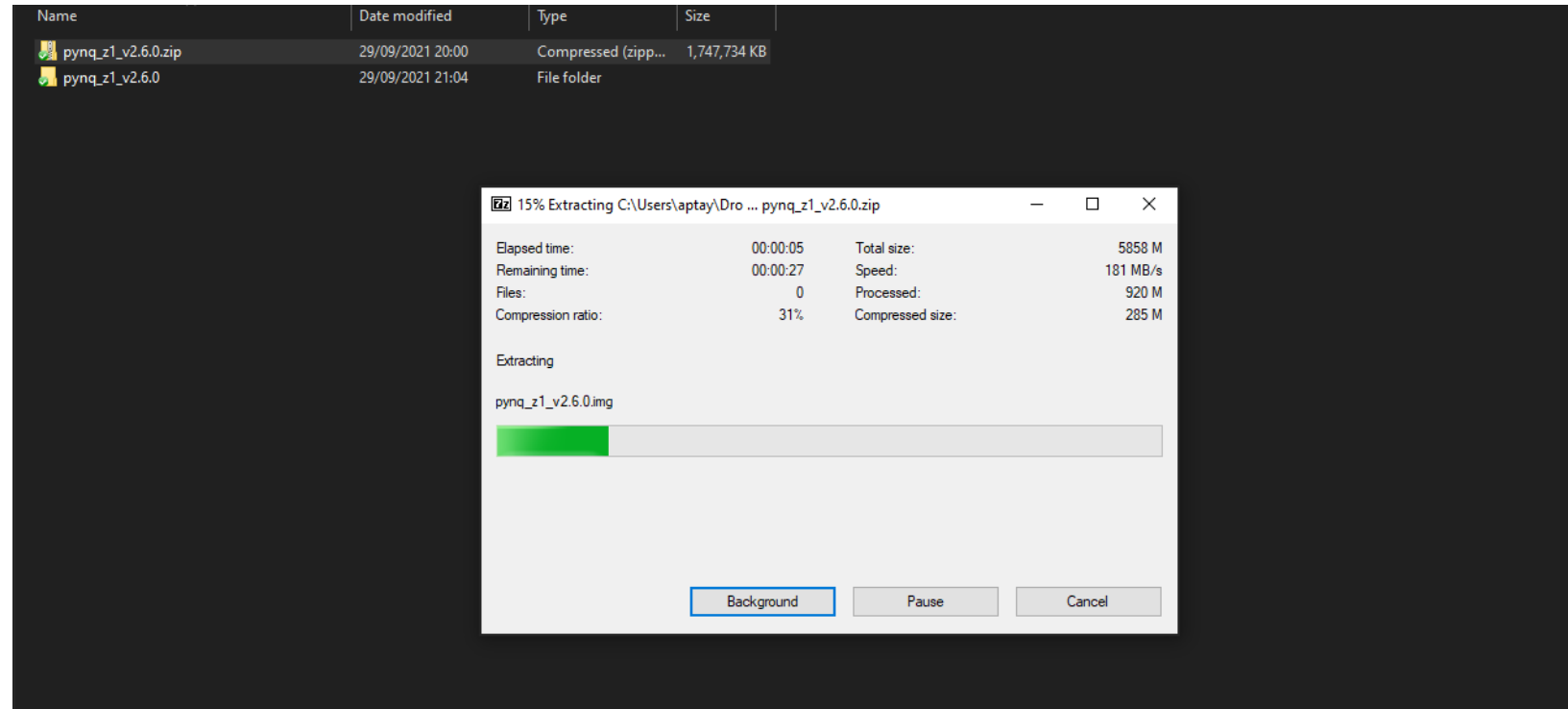
Lab : FFT Acceleration

Let Vivado manage the wrapper



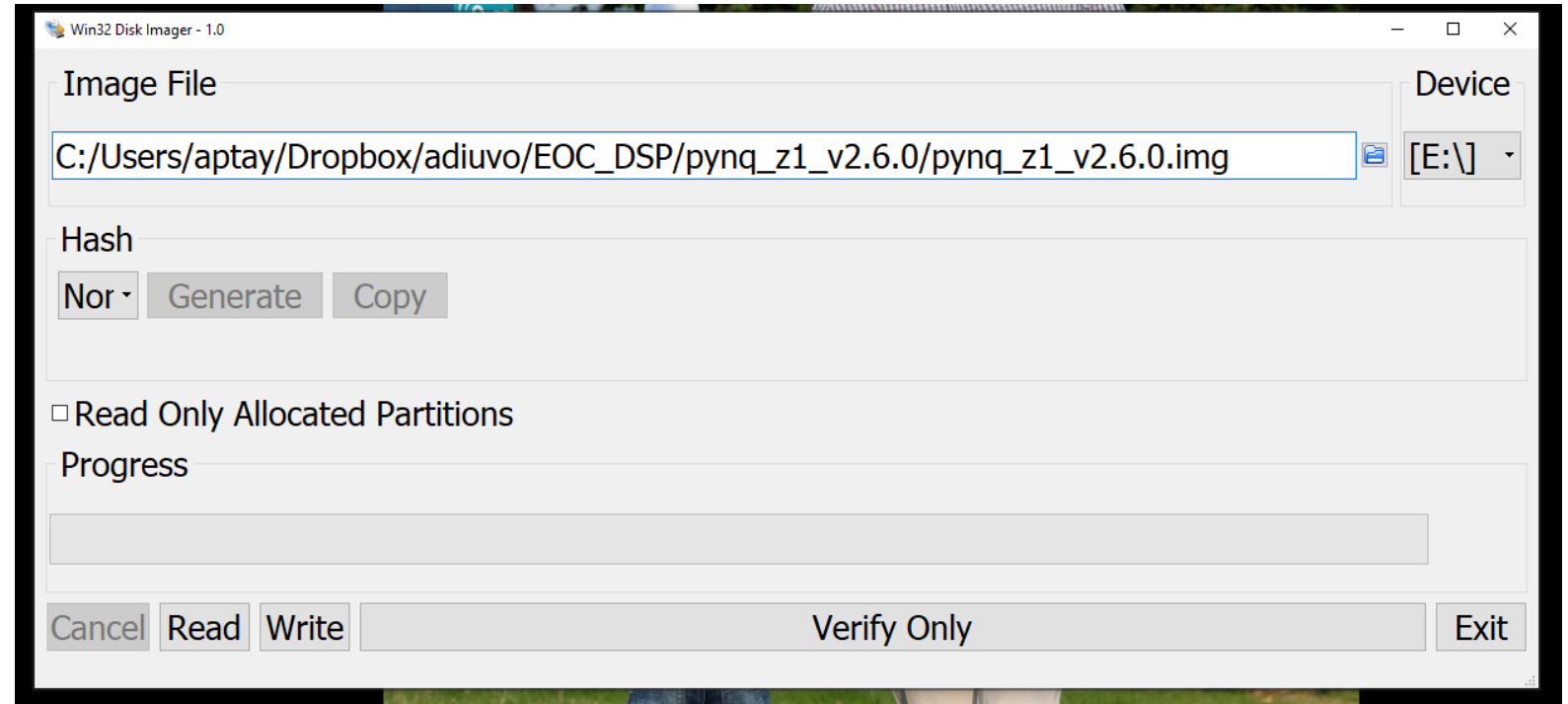
Lab : FFT Acceleration

Extract the
downloaded PYNQ
image



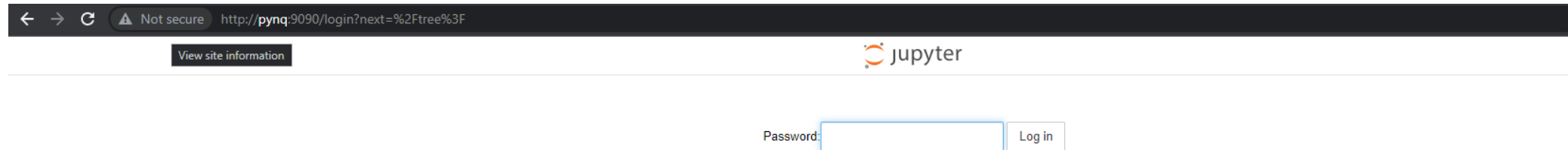
Lab : FFT Acceleration

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



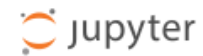
Lab : FFT Acceleration

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 Acceleration

Once logged in you should see the folder structure below

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

Select items to perform actions on them.

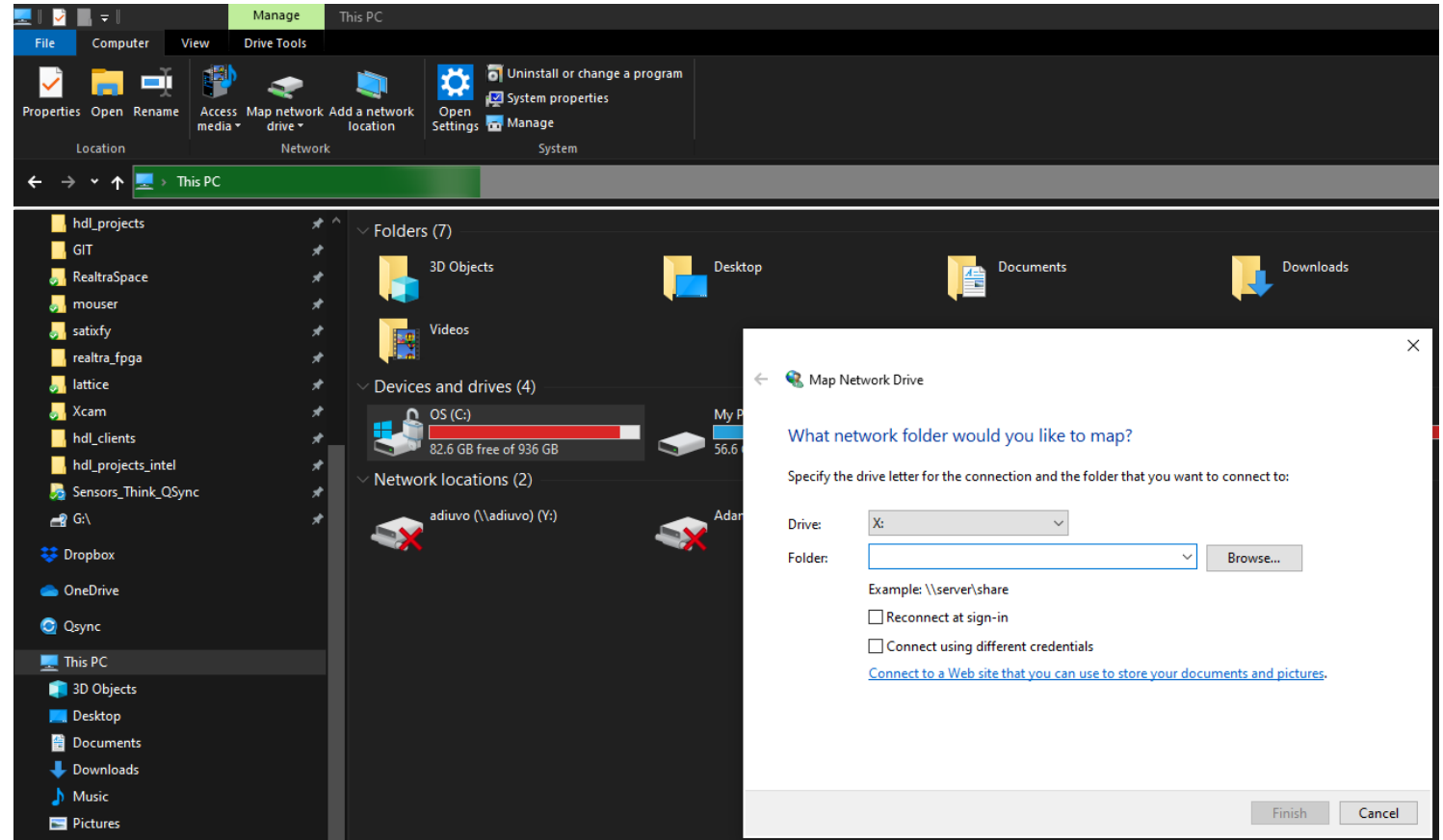
[Upload](#)[New ▾](#)☐ 0 ▾[Name ▾](#)[Last Modified](#)☐[base](#)[a year ago](#)☐[common](#)[a year ago](#)☐[getting_started](#)[a year ago](#)☐[logictools](#)[a year ago](#)☐[Welcome to Pynq.ipynb](#)[a year ago](#)

Lab : FFT Acceleration

In a file explorer map a
network drive to

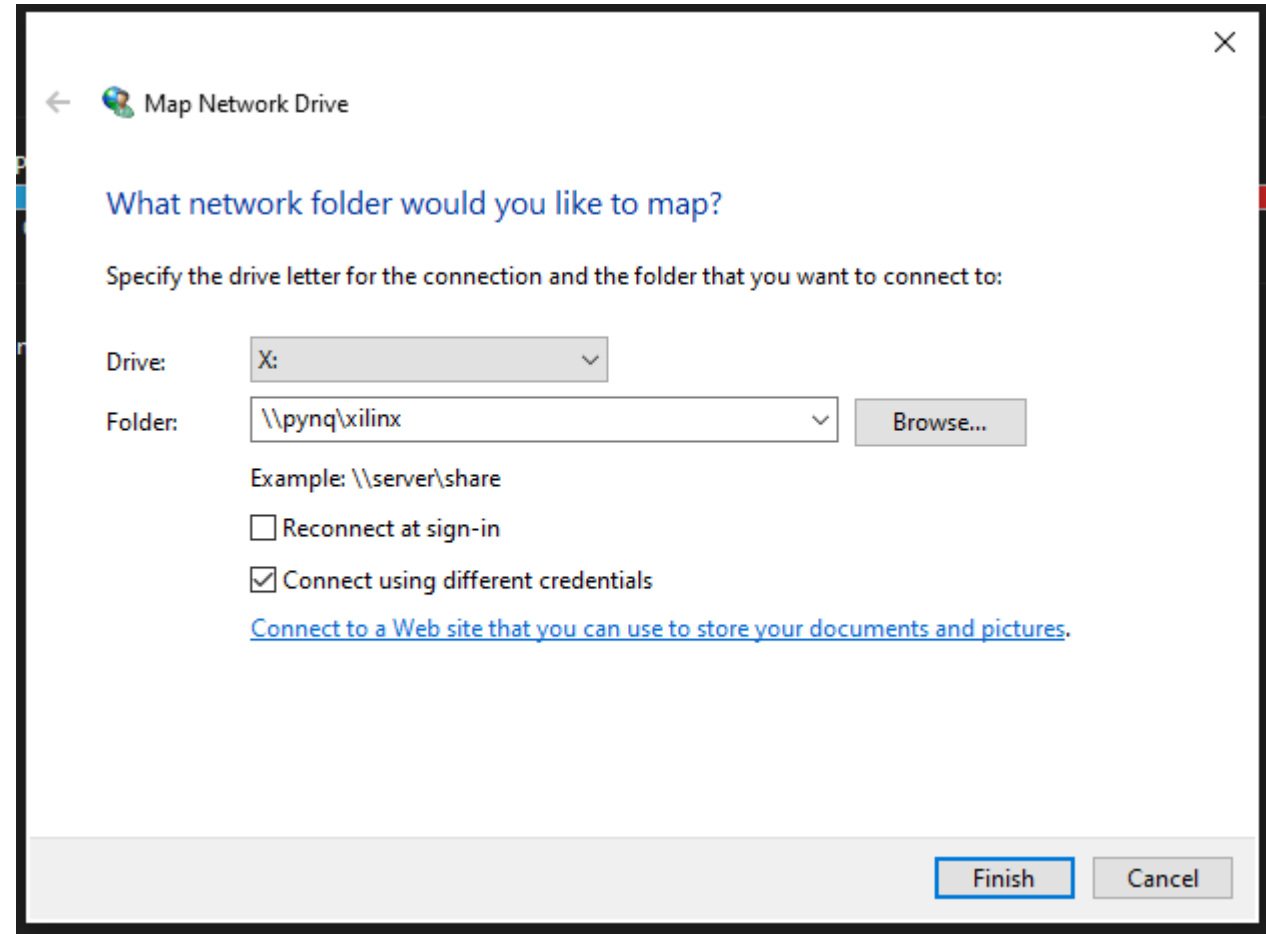
`\\pynq \xilinx`

Select connect using different
credentials



Lab : FFT Acceleration

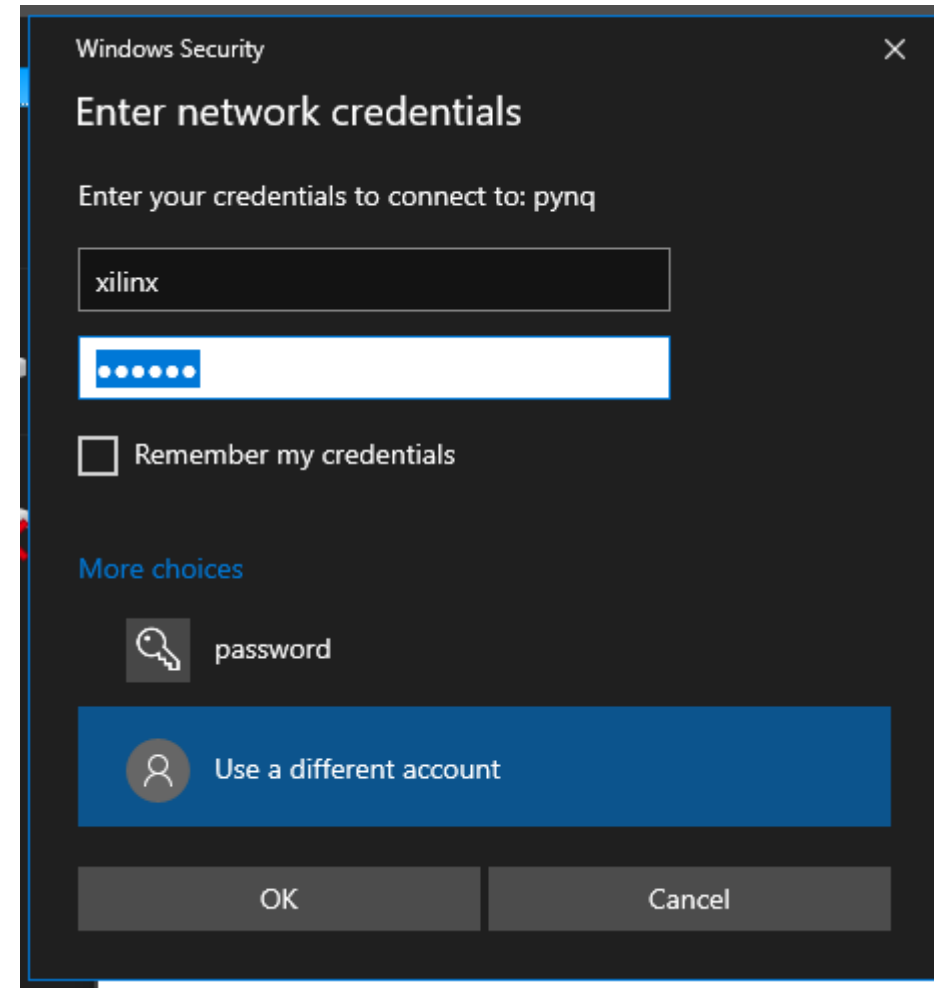
Completed Map drive, click OK



Lab : FFT Acceleration

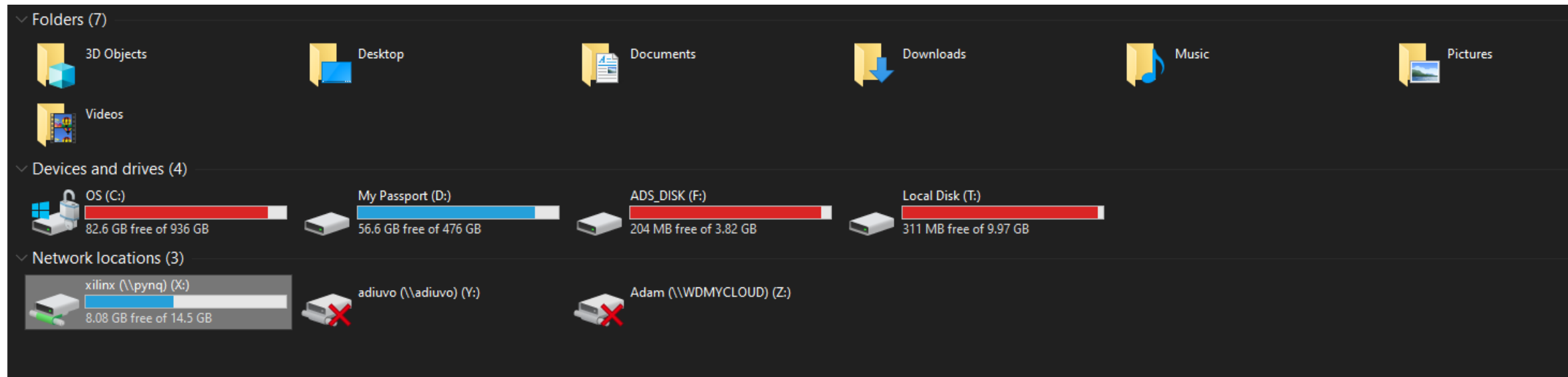
Enter the username and password as

Xilinx click OK



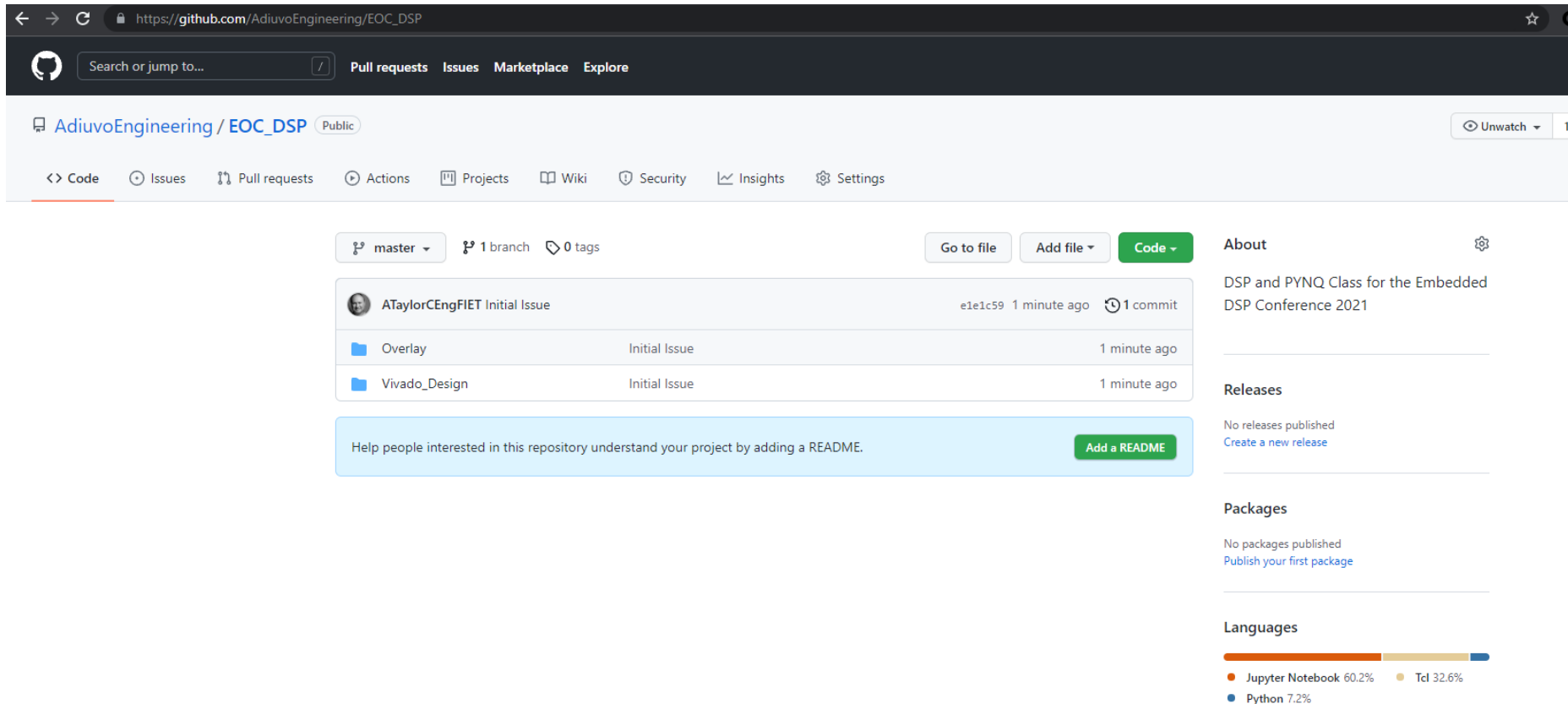
Lab : FFT Acceleration

The pynq drive should not appear as a samba server



Lab : FFT Acceleration

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 "Releases" (no releases published) and "Packages" (no packages published). The "Languages" section shows the following distribution: Jupyter Notebook (60.2%), Tcl (32.6%), and Python (7.2%).

https://github.com/AdiuvoEngineering/EOC_DSP

Search or jump to... Pull requests Issues Marketplace Explore

AdiuvoEngineering / EOC_DSP Public Unwatch 1

<> Code Issues Pull requests Actions Projects Wiki Security Insights Settings

master 1 branch 0 tags Go to file Add file Code

ATaylorCEngFIET Initial Issue e1e1c59 1 minute ago 1 commit

File	Initial Issue	1 minute ago
Overlay	Initial Issue	1 minute ago
Vivado_Design	Initial Issue	1 minute ago

Help people interested in this repository understand your project by adding a README. Add a README

About DSP and PYNQ Class for the Embedded DSP Conference 2021

Releases No releases published Create a new release

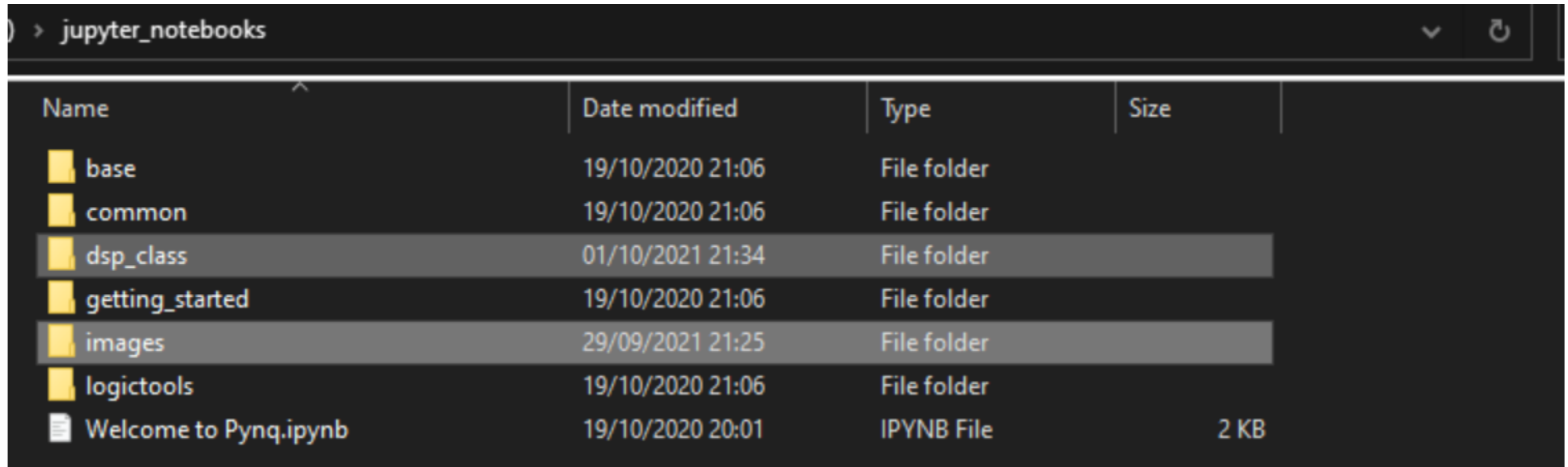
Packages No packages published Publish your first package

Languages

- Jupyter Notebook 60.2%
- Tcl 32.6%
- Python 7.2%

Lab : FFT Acceleration

From the Cloned Repo copy the directory Images and dsp_class to the Pynq boards Jupyter notebooks directory



jupyter_notebooks				
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 Acceleration

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 Acceleration

Select fft.ipynb it will open and start running

 jupyter Logout

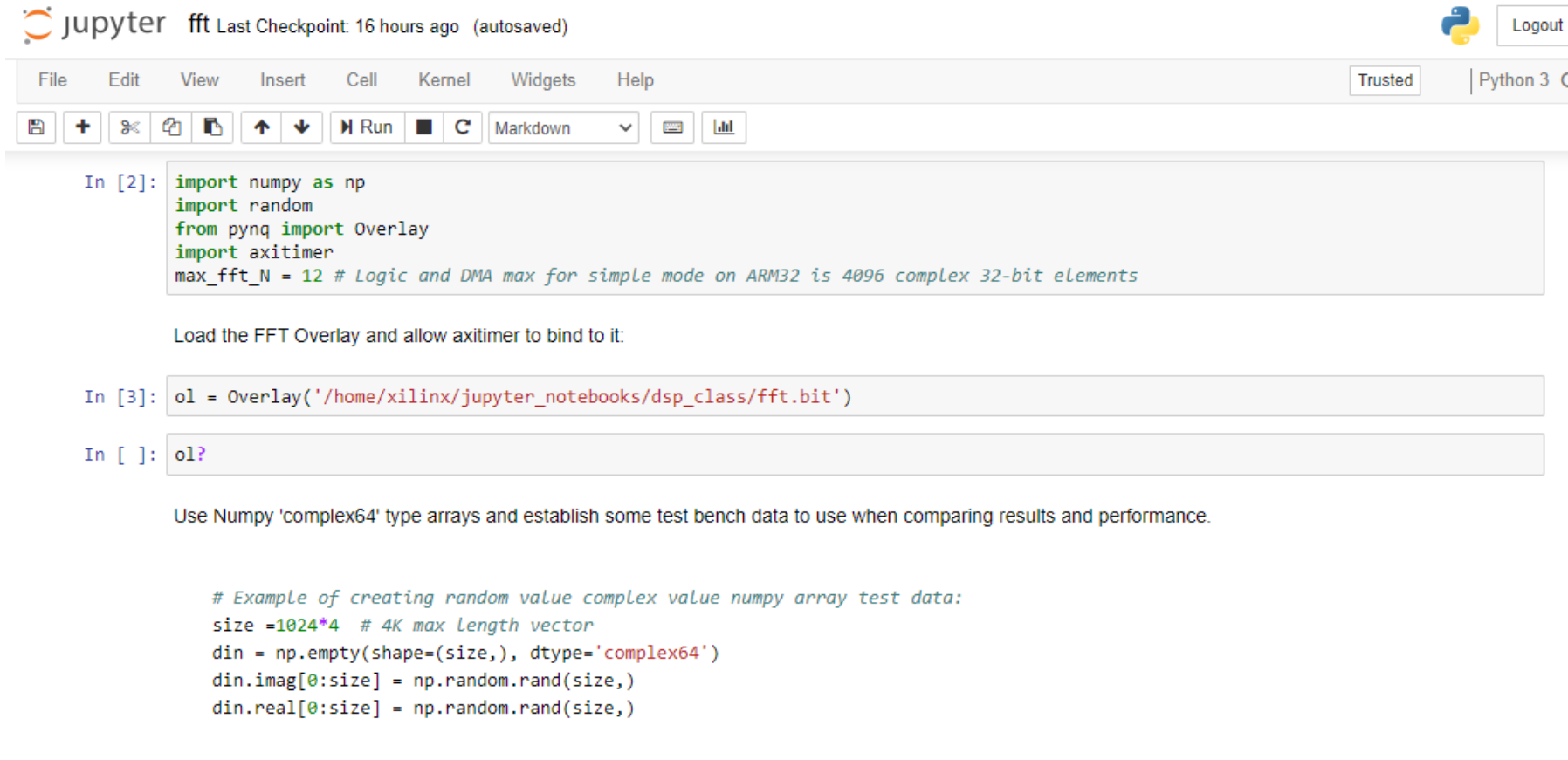
Files Running Clusters Nbextensions

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 Acceleration

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 title "fft Last Checkpoint: 16 hours ago (autosaved)". The interface includes a menu bar (File, Edit, View, Insert, Cell, Kernel, Widgets, Help), a toolbar with icons for saving, adding cells, undo, redo, and running, and a status bar showing "Trusted" and "Python 3". The notebook contains three code cells:

```
In [2]: import numpy as np
import random
from pyq 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,)
```



ADIUVO

ENGINEERING AND TRAINING, LTD.

www.adiuvoengineering.com



adam@adiuvoengineering.com