ADIUVO

ENGINEERING AND TRAINING, LTD.

Signal Processing with FPGA, Python & no RTL Design!

Adam Taylor

Adam@AdiuvoEngineering.com

ADIUVO
ENGINEERING AND TRAINING, LTD.

Let's Use an FPGA

Project Managers Despair

- Timescales Increasing
- Costs Exploding
- Difficult conversations with engineering
- Even more difficult conversations with senior management.

Of course, it does not have to be this way!

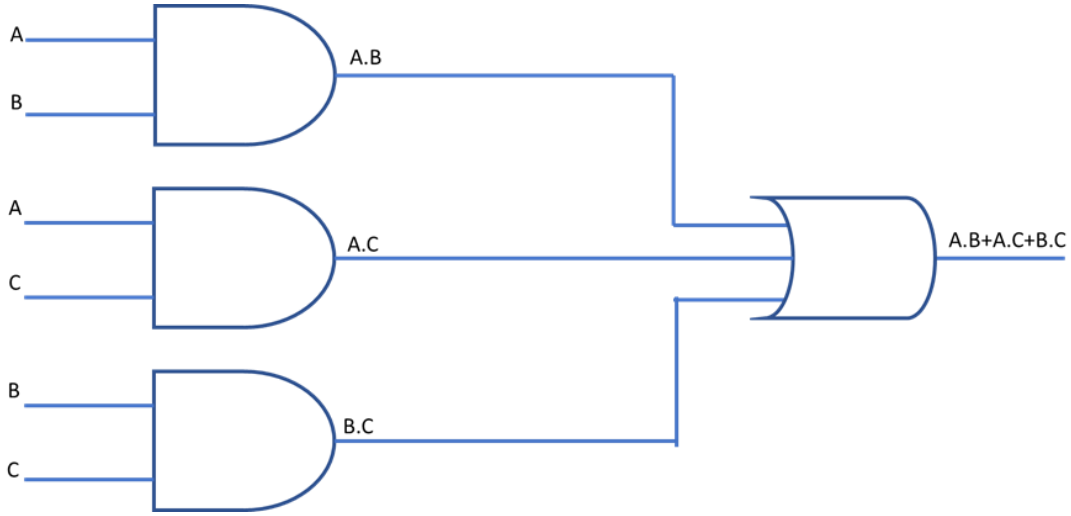
FPGA

ADIUVO
ENGINEERING AND TRAINING, LTD.

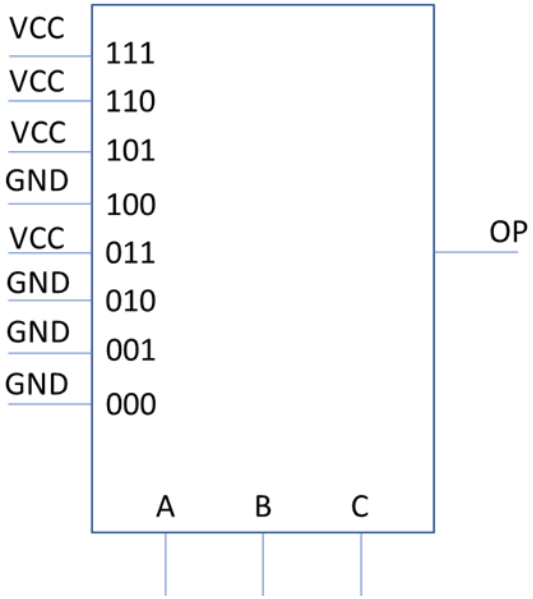
ADIUVO
ENGINEERING AND TRAINING, LTD.

FPGA Architecture

What is programmable logic



Input A	Input B	Input C	OP
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

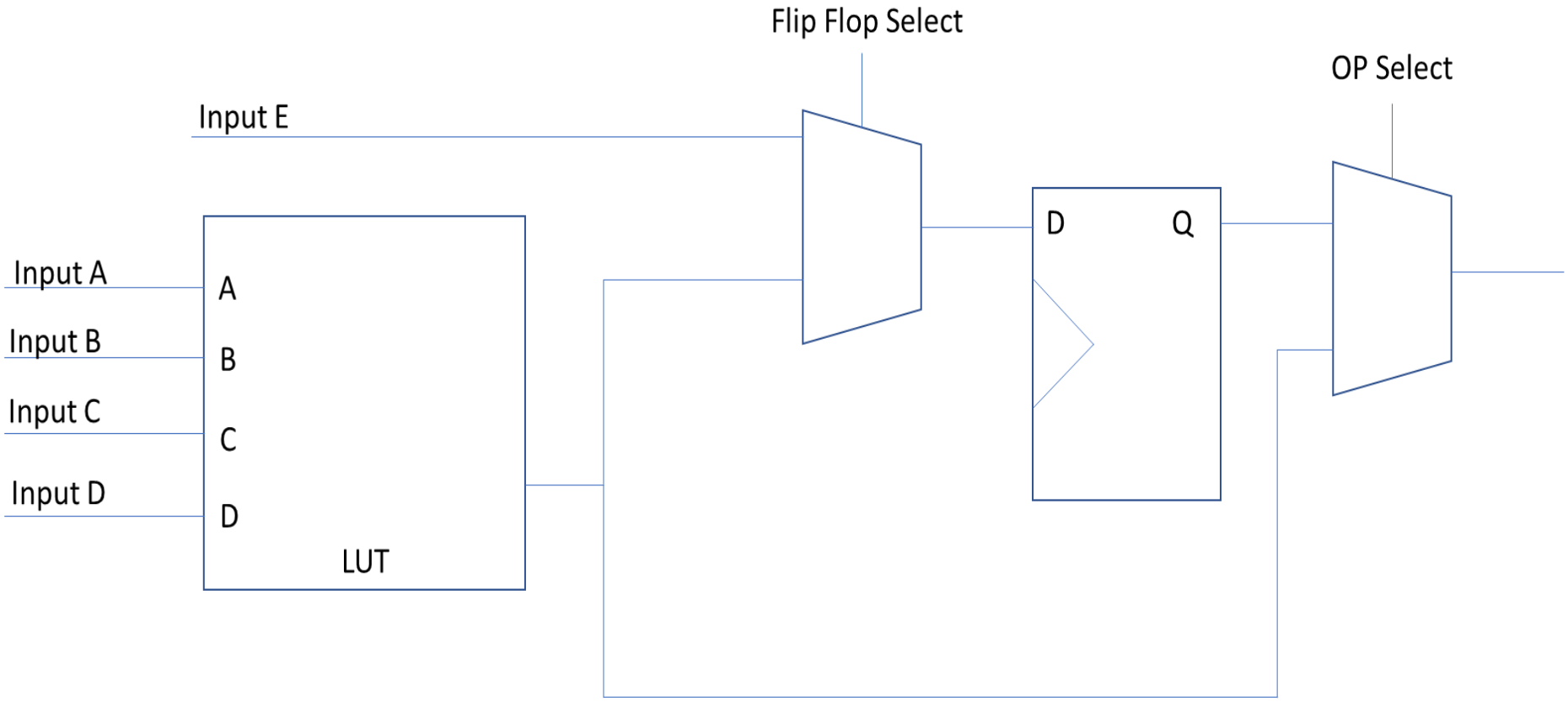


ADIUVO
ENGINEERING AND TRAINING, LTD.

Configurable Logic Block

- LUT allows implementation of Combinatorial Circuit
- BUT we design synchronous circuits what about the FF
- flip flop to act as storage for the combinatorial output such that we can implement sequential structures
- This combination of a LUT and Flip Flop is often called a Configurable Logic Block (CLB) and a programmable logic device will consist of many thousands of these CLBs. To provide the most flexibility additional multiplexers will be used to support a wide range of CLB configurations.

Configurable Logic Block

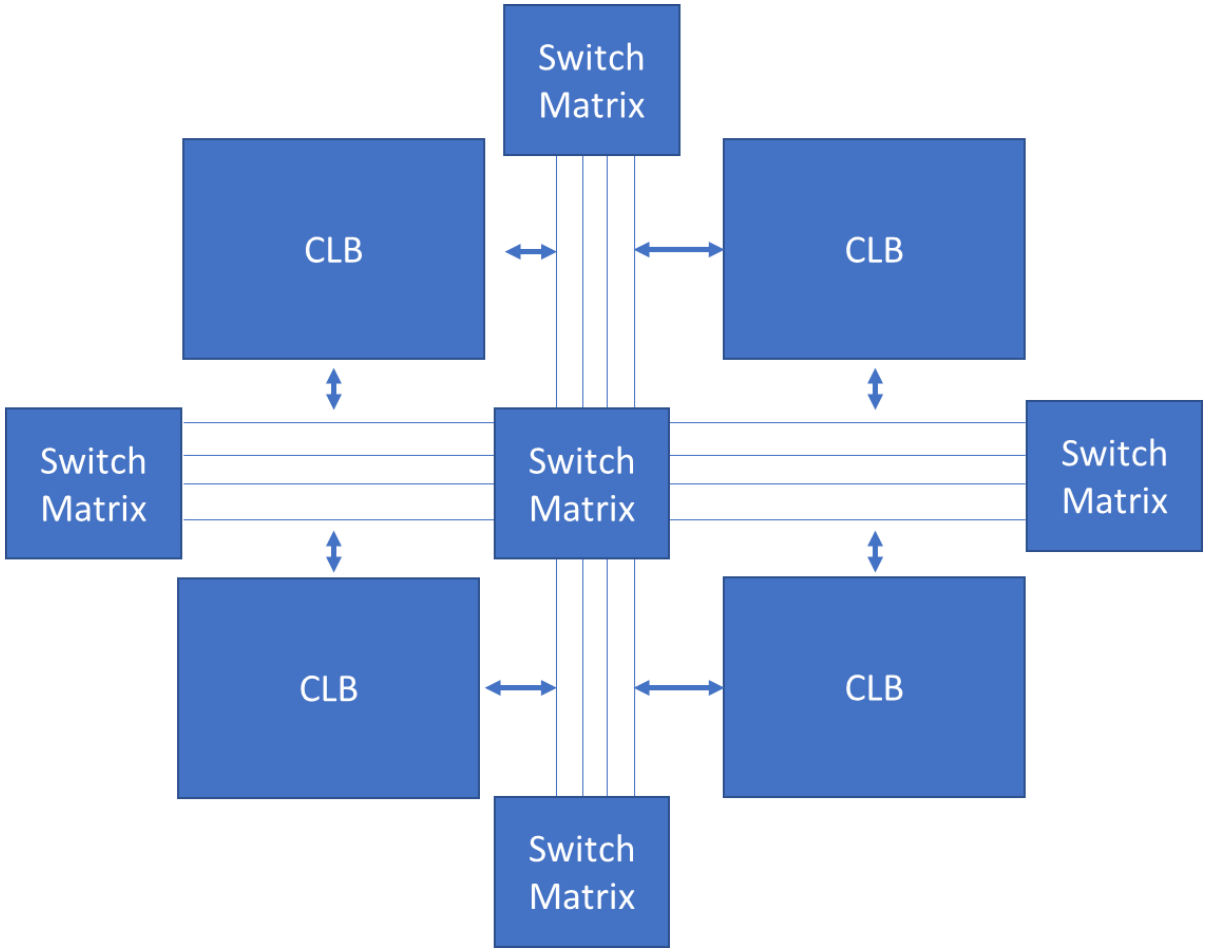


Interconnect Routing

A collection of wires and programmable switches.

These are responsible for connecting CLBs and other building blocks within the FPGA.

These are also called routing channels.



Heterogeneous SoC

Heterogeneous SOC

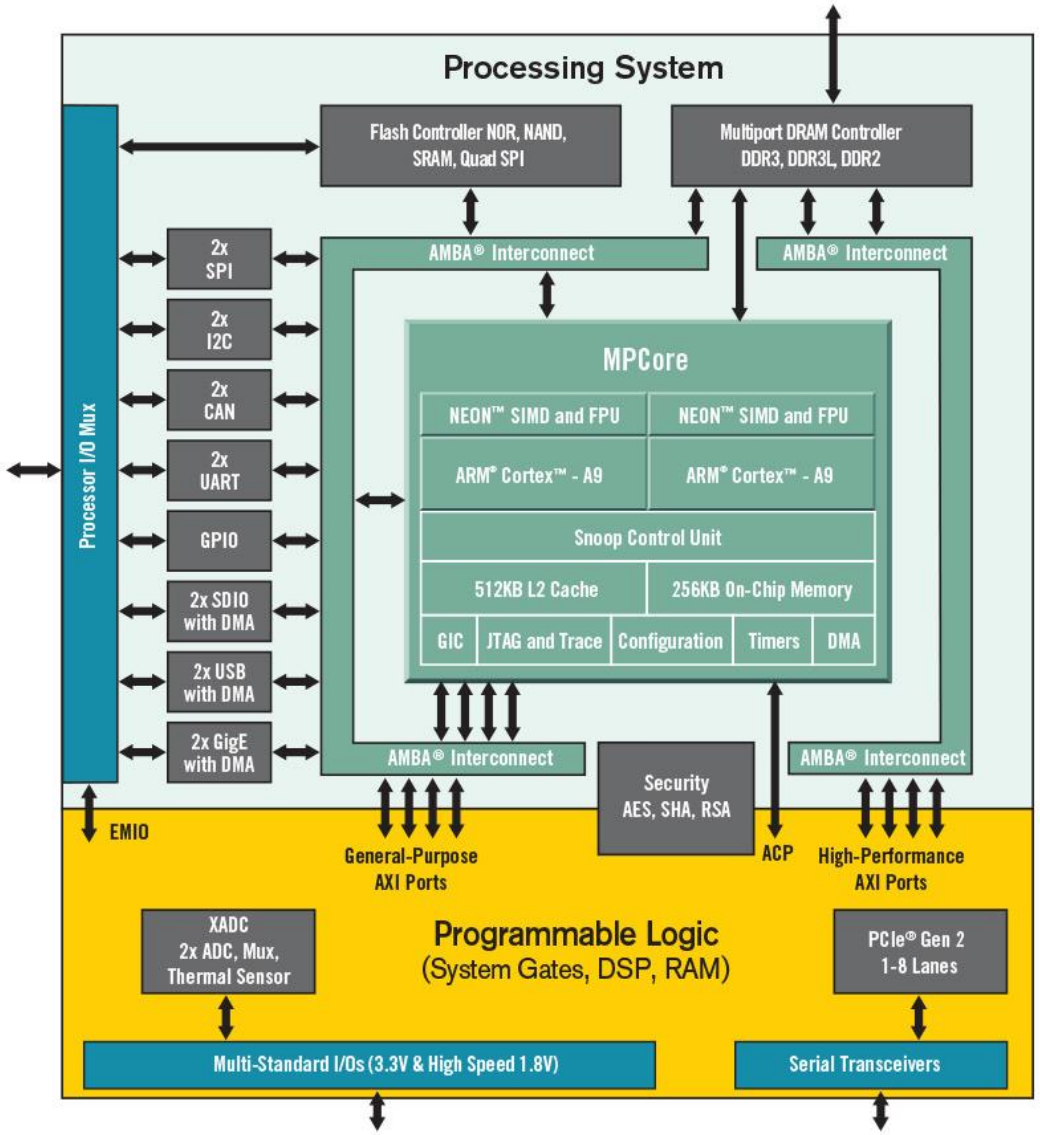
Combine diverse processing elements with programmable logic

Processing elements are silicon implementations

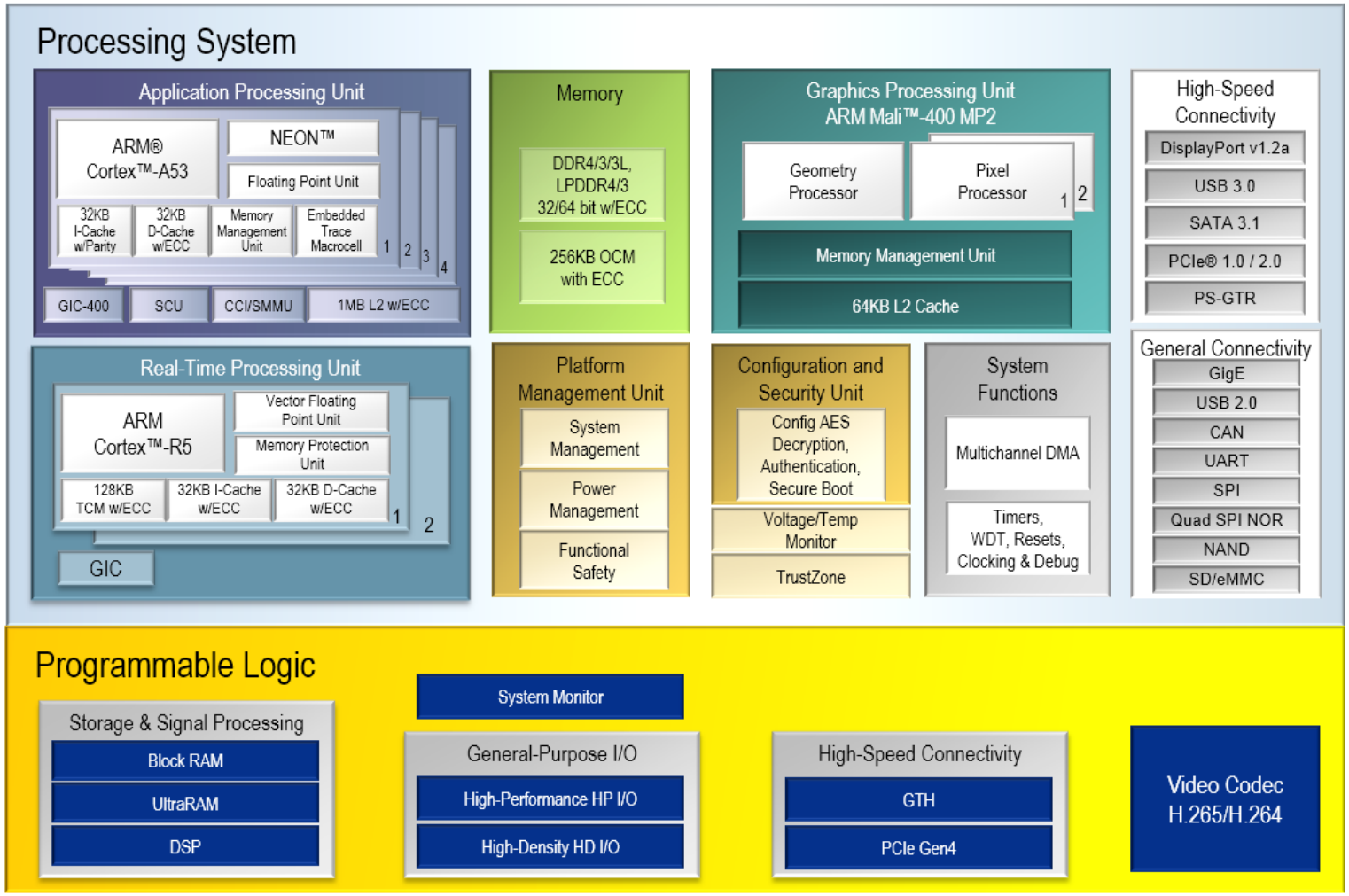
Processing element is the master – it boots just like any other processor

Enables highly optimized solutions be implemented using processor and logic to exploit natural strengths

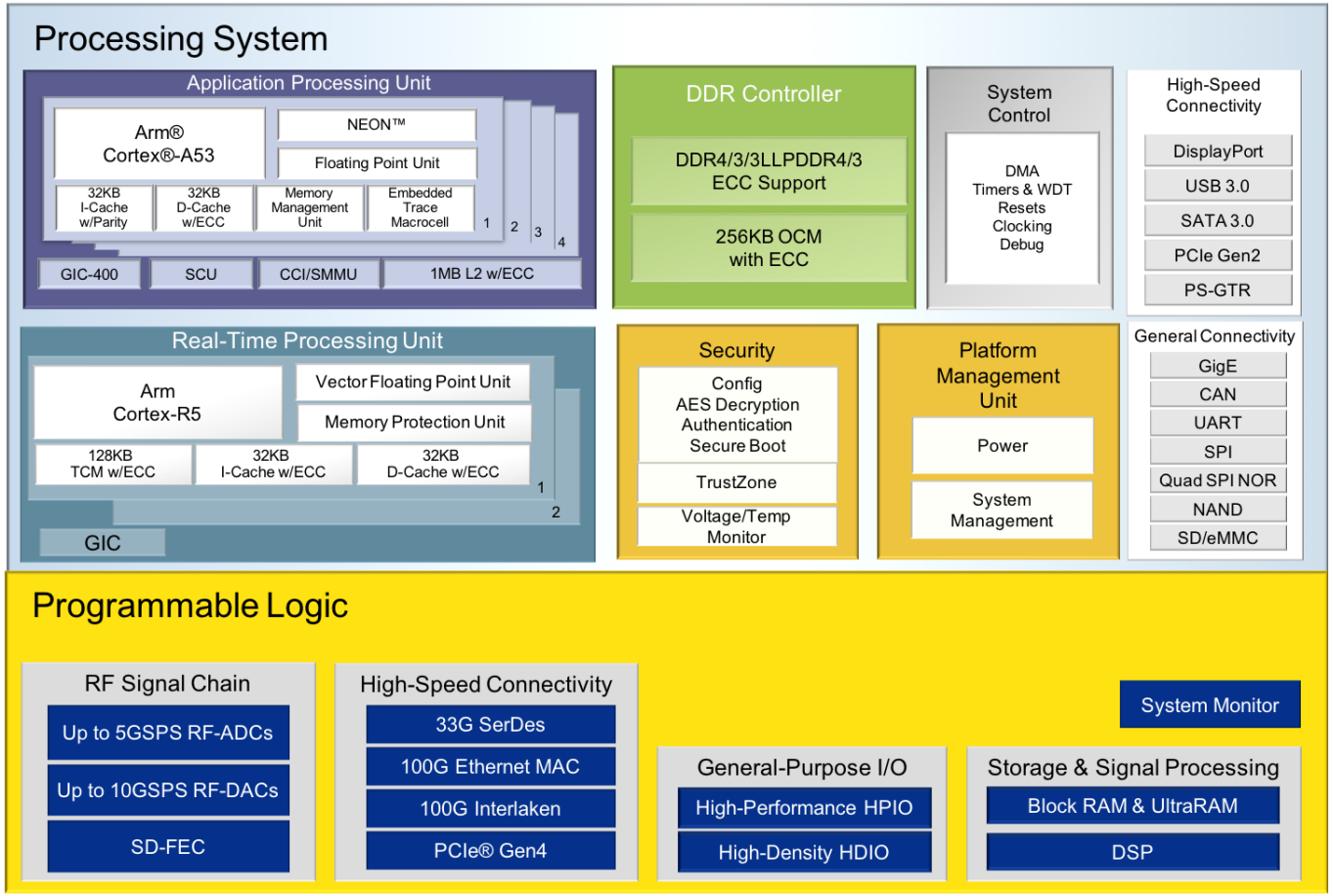
Zynq



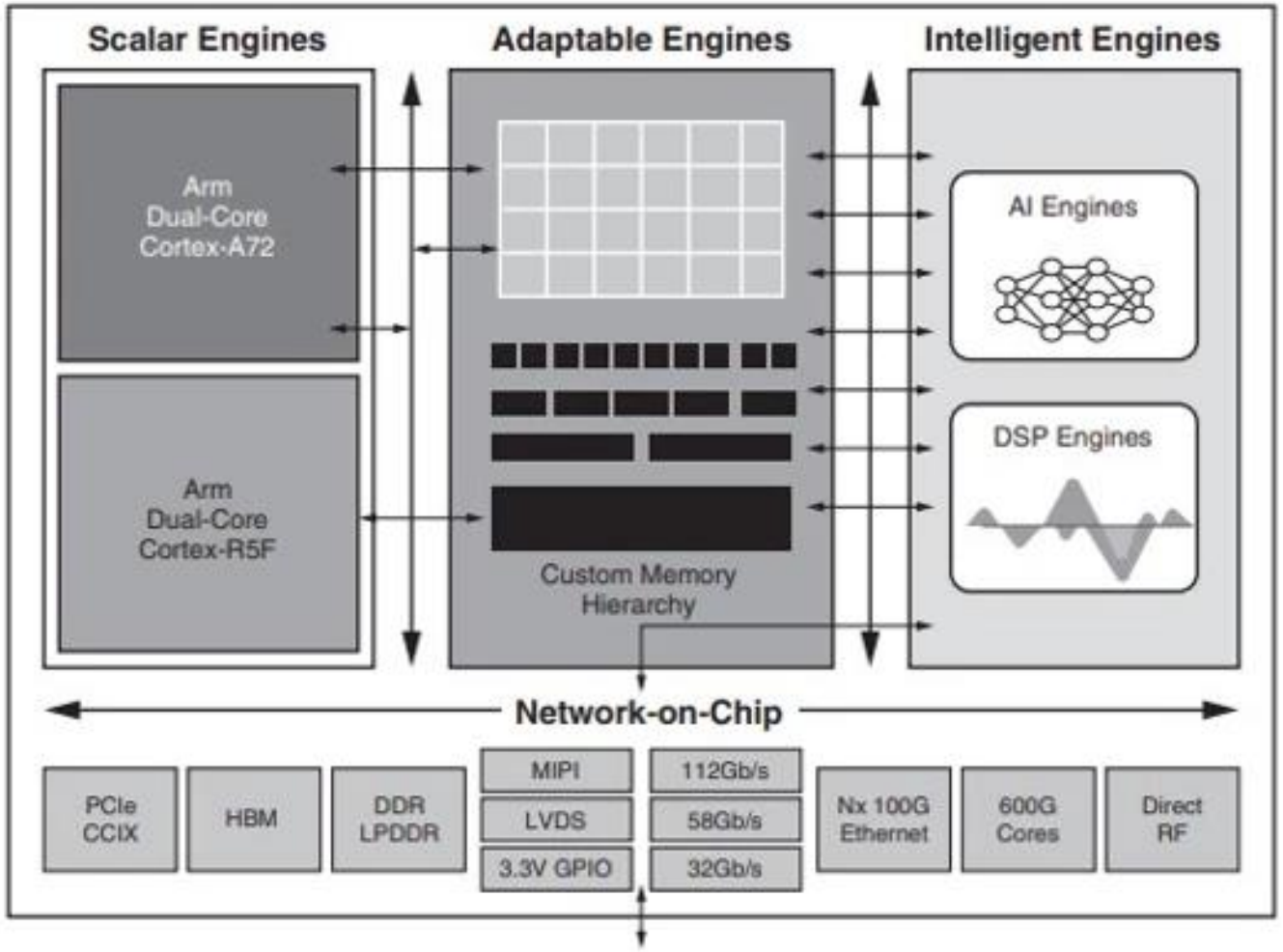
Zynq MPSoC



Zynq RFSoC

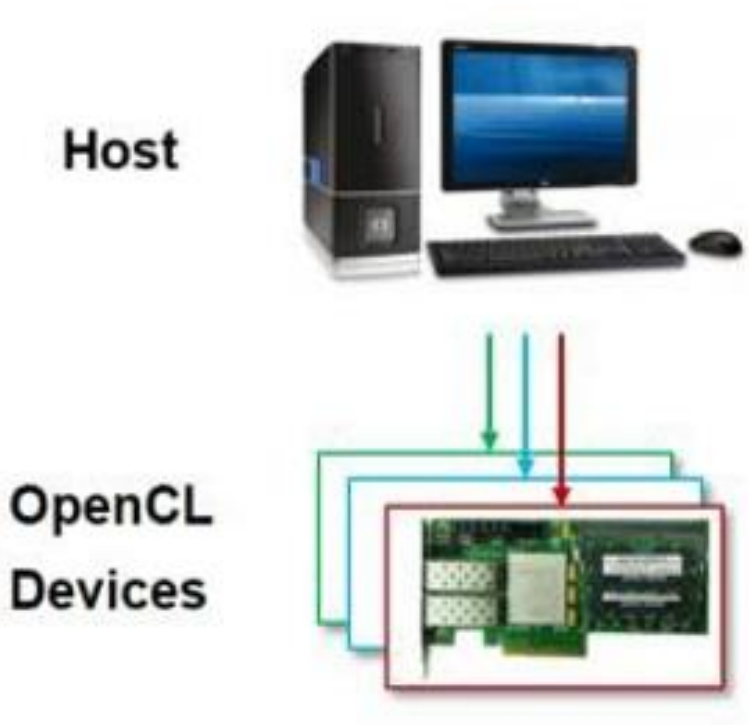


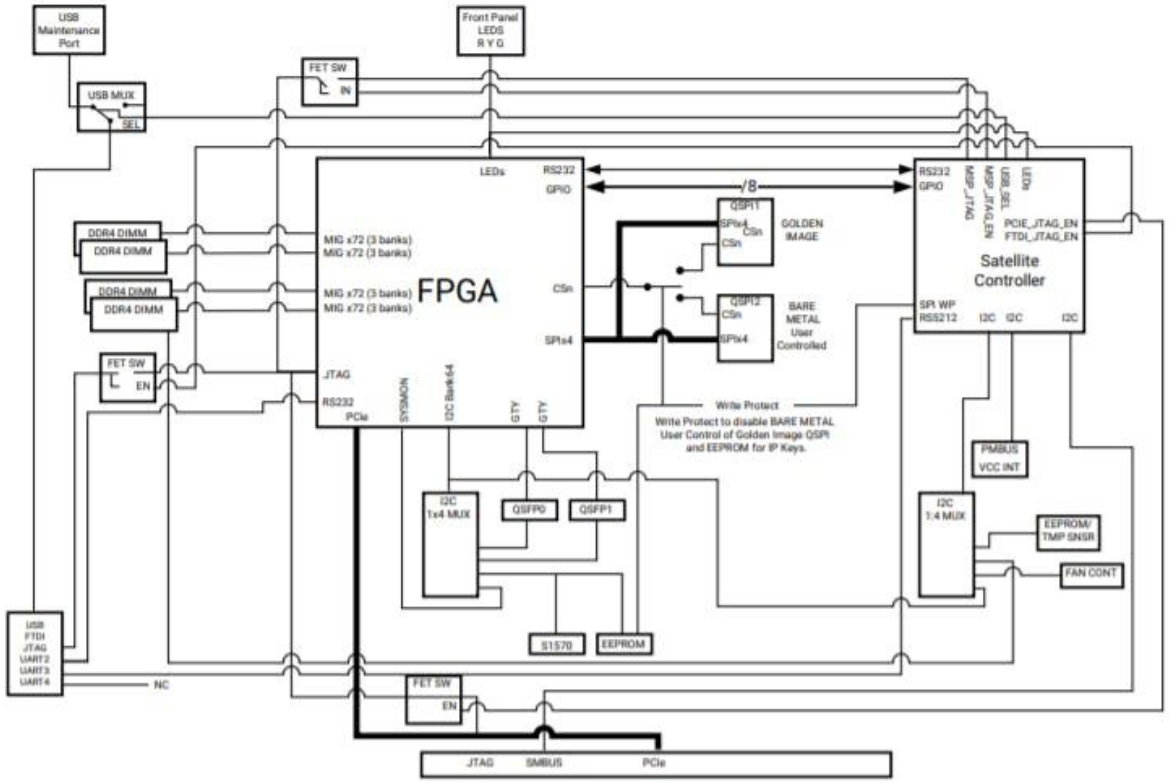
Versal



Alveo

Accelerator card based around OpenCL





Tool Chain

ADIUVO
ENGINEERING AND TRAINING, LTD.

Xilinx Tools and Frameworks

VitisAI

Enables the implementations of machine learning inference, using Tensor Flow, Caffe and PyTorch. *Requires a SoC/RFSoC/Alveo

Vitis

Embedded and accelerated SW development. Used to develop software solutions for MicroBlaze, Arm R5, A9, A53 and A72.

Petalinux

Embedded Linux solutions

PYNQ

Python framework for rapid prototyping on SoC/RFSoC/Alveo

Vitis / Vivado HLS

High Level Synthesis tool supporting C/C++/OpenCL

Vivado

Design Capture and implementation for the base platform

Vivado Overview

- Foundation of all design and higher-level tools is Vivado
- Vivado enables us to capture designs using VHDL or Verilog
- Large IP library to accelerate our designs
- Integrates Vivado / Vitis HLS IP cores

- ✓ Simulate designs using Vivado Simulator
- ✓ Synthesize, place and route the design
- ✓ Generate power estimations
- ✓ Create Xilinx Support Architecture

Vivado Overview

FPGA Implementation Flow

Synthesis – Translates the HDL design into a series of logic equations which are then mapped onto the resources available in the target FPGA.

Synthesis – Place – The logic resources determined by the synthesis tool are placed at available locations within the target device.

Routing – The placed logic resources in the design are interconnected using routing and switch matrixes to implement the final application.

Bit File – The generation of the final programming file for the target FPGA.

We can control the flow implementation settings by using Constraints (XDC file) and implementation Strategies

Vivado IP Integrator

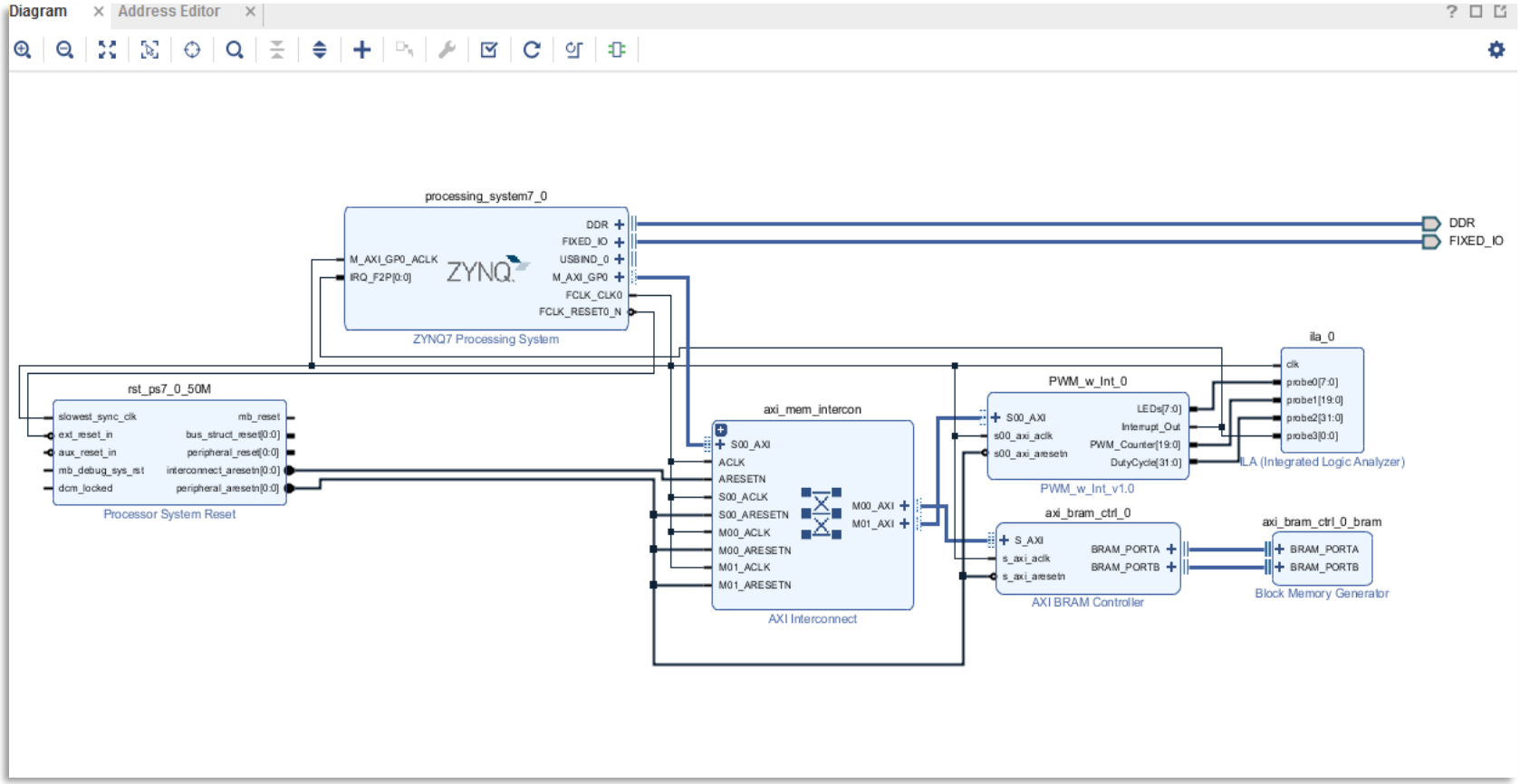
ADIUVO
ENGINEERING AND TRAINING, LTD.

Vivado IP Integrator (IPI)

Create system-level designs

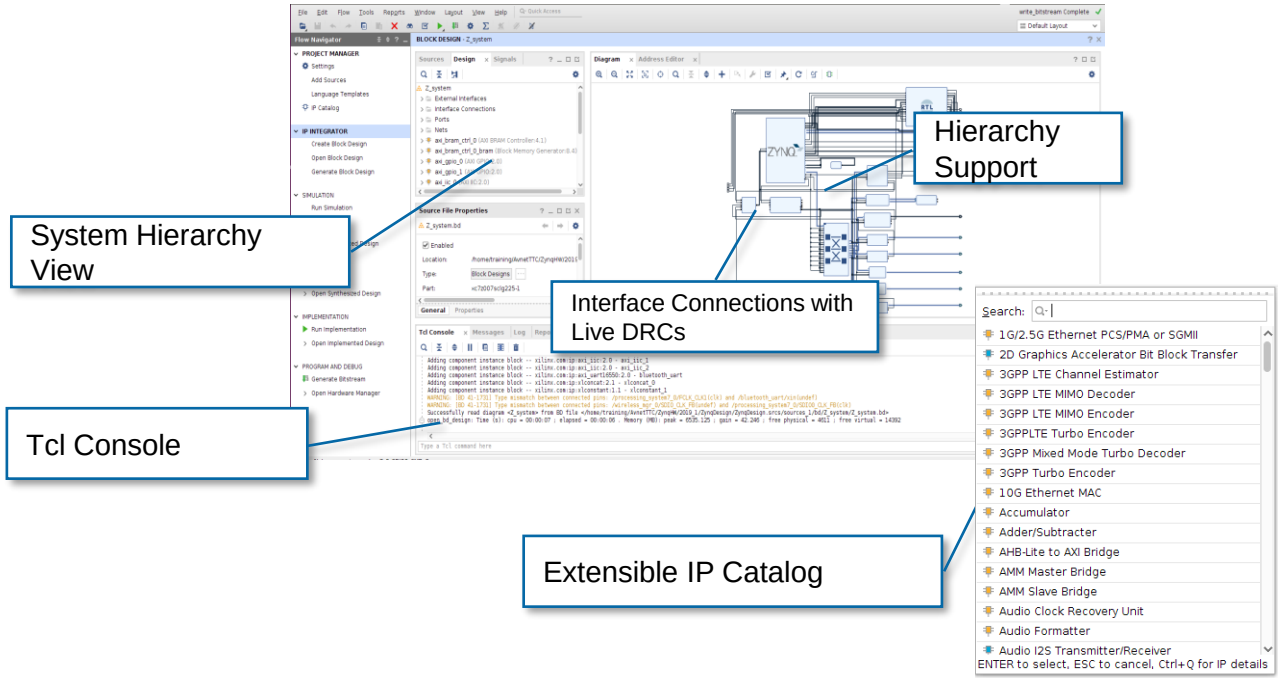
- Instantiate and interconnect IP cores

- IP-centric design flow
  - Plug-and-play IP
  - Vast IP catalog
- Accelerates
  - Integration
  - Productivity
- Example applications
  - Embedded
  - DSP
  - Video
  - Analog
  - Networking



Vivado IP Integrator: Intelligent IP Integration

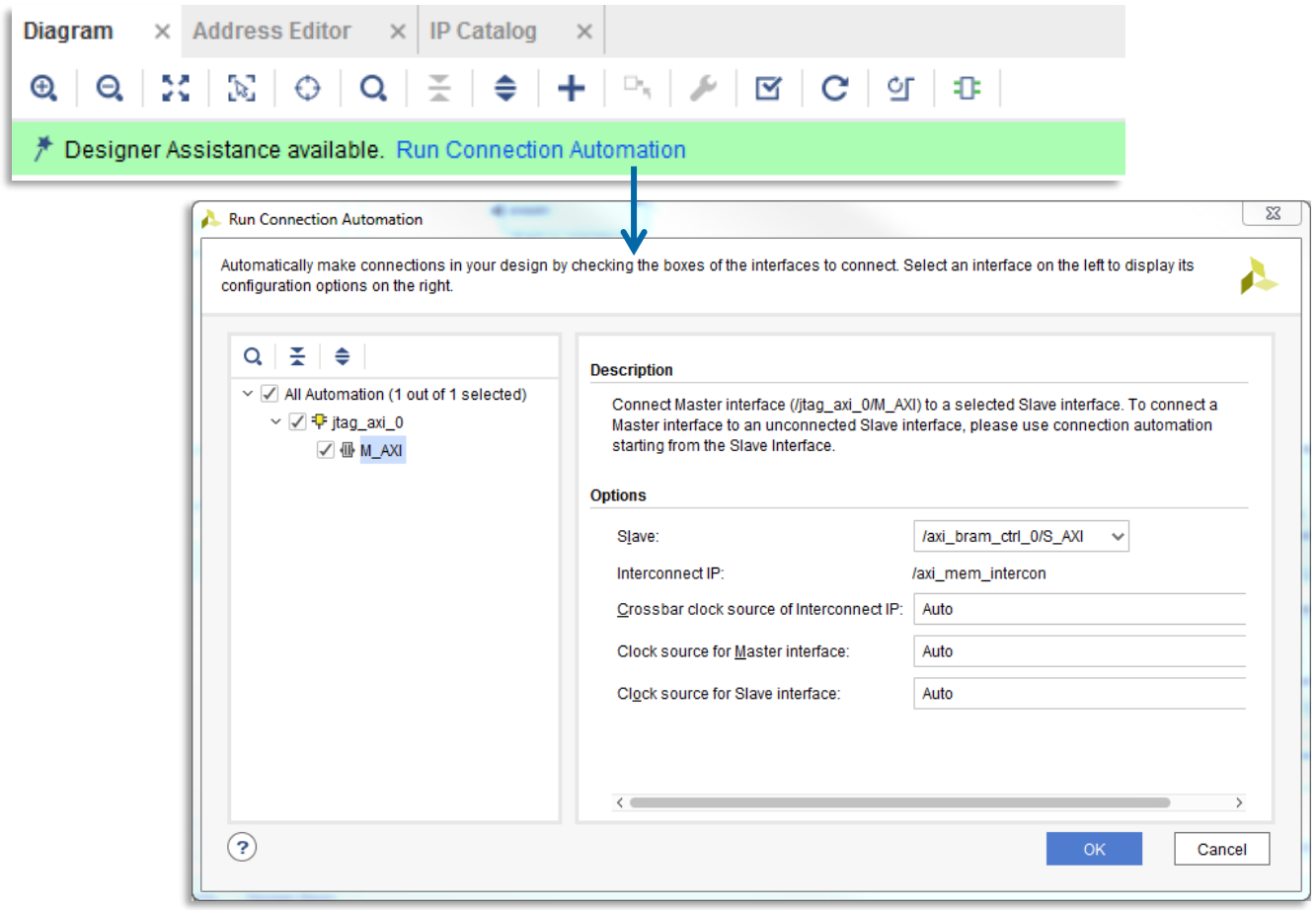
- Automated IP subsystems
- Block automation for rapid design creation
- One click IP customization
- Board aware
- Support all 7 Series FPGAs and Zynq SoCs
- Built-in presets, accelerating design creation



Vivado IP Integrator: Intelligent IP Integration

Correct-by-construction

- Interface level connections
- Extensible IP repository
- Real-time DRCs and parameter propagation / resolution
- Designer assistance



IP Integrator Library

Error Correction				
3GPPLTE Turbo Encoder		Production	Purchase	xilinx.co
3GPP Mixed Mode Turbo Decoder	AXI4-Stream	Production	Purchase	xilinx.co
3GPP Turbo Encoder		Production	Purchase	xilinx.co
Convolution Encoder	AXI4-Stream	Production	Included	xilinx.co
Fibre Channel 32GFC RS-FEC			Purchase	xilinx.co
FlexO 100G RS-FEC	AXI4		Purchase	xilinx.co
G.709 FEC Encoder/Decoder	AXI4, AXI4-Stream	Not-Supported	Purchase	xilinx.co
G.709 FEC Encoder/Decoder	AXI4, AXI4-Stream	Not-Supported	Purchase	xilinx.co
G.975.1 EFEC I.4 Encoder/Decoder	AXI4-Stream	Not-Supported	Purchase	xilinx.co
G.975.1 EFEC I.7 Encoder/Decoder	AXI4-Stream	Not-Supported	Purchase	xilinx.co
IEEE 802.3 25G RS-FEC	AXI4		Purchase	xilinx.co
IEEE 802.3 50G RS-FEC	AXI4		Purchase	xilinx.co
IEEE 802.3 50G RS-FEC	AXI4		Purchase	xilinx.co
IEEE 802.3 400G RS-FEC			Purchase	xilinx.co
IEEE 802.3bj 100G RS-FEC	AXI4		Purchase	xilinx.co
IEEE 802.3bj 100G RS-FEC	AXI4		Purchase	xilinx.co
IEEE 802.3 Clause 74 FEC			Purchase	xilinx.co
IEEE 802.3 Multi-channel 25G RSFEC			Purchase	xilinx.co
Interleaver/De-interleaver	AXI4-Stream	Production	Purchase	xilinx.co
LDPC Encoder/Decoder	AXI4, AXI4-Stream	Production	Purchase	xilinx.co
Polar Encoder/Decoder	AXI4, AXI4-Stream	Production	Purchase	xilinx.co
Reed-Solomon Decoder	AXI4-Stream	Production	Purchase	xilinx.co
Reed-Solomon Encoder	AXI4-Stream	Production	Purchase	xilinx.co
Soft-Decision FEC	AXI4, AXI4-Stream		Purchase	xilinx.co
Viterbi Decoder	AXI4-Stream	Production	Purchase	xilinx.co

IP Integrator Library

Wireless				
3GPP LTE Channel Estimator	AXI4-Stream	Production	Purchase	xilinx.cc
3GPP LTE MIMO Decoder		Production	Purchase	xilinx.cc
3GPP LTE MIMO Encoder	AXI4-Stream	Production	Purchase	xilinx.cc
CPRI	AXI4, AXI4-Stream		Purchase	xilinx.cc
DUC/DDC Compiler	AXI4-Stream	Production	Included	xilinx.cc
JESD204	AXI4, AXI4-Stream		Purchase	xilinx.cc
JESD204C	AXI4, AXI4-Stream		Purchase	xilinx.cc
JESD204 PHY	AXI4		Included	xilinx.cc
LDPC Encoder/Decoder	AXI4, AXI4-Stream	Production	Purchase	xilinx.cc
LTE DL Channel Encoder	AXI4, AXI4-Stream	Production	Purchase	xilinx.cc
LTE Fast Fourier Transform		Production	Purchase	xilinx.cc
LTE PUCCH Receiver	AXI4-Stream	Production	Purchase	xilinx.cc
LTE UL Channel Decoder	AXI4, AXI4-Stream	Production	Purchase	xilinx.cc
Peak Cancellation Crest Factor Reduction	AXI4, AXI4-Stream	Production	Purchase	xilinx.cc
Polar Encoder/Decoder	AXI4, AXI4-Stream	Production	Purchase	xilinx.cc
Serial RapidIO Gen2	AXI4, AXI4-Stream		Purchase	xilinx.cc
Soft-Decision FEC	AXI4, AXI4-Stream		Purchase	xilinx.cc

IP Integrator Library

Digital Signal Processing				
Building Blocks				
Complex Multiplier	AXI4-Stream	Production	Included	xilinx.co
CORDIC	AXI4-Stream	Production	Included	xilinx.co
Filters				
CIC Compiler	AXI4-Stream	Production	Included	xilinx.co
DUC/DDC Compiler	AXI4-Stream	Production	Included	xilinx.co
FIR Compiler	AXI4-Stream	Production	Included	xilinx.co
Modulation				
DDS Compiler	AXI4-Stream	Production	Included	xilinx.co
Transforms				
DFTs				
Discrete Fourier Transform		Production	Included	xilinx.co
FFTs				
Fast Fourier Transform	AXI4-Stream	Production	Included	xilinx.co
LTE Fast Fourier Transform		Production	Purchase	xilinx.co
Trig Functions				
CORDIC	AXI4-Stream	Production	Included	xilinx.co
DDS Compiler	AXI4-Stream	Production	Included	xilinx.co
Waveform Synthesis				
DDS Compiler	AXI4-Stream	Production	Included	xilinx.co

What is PYNQ – Introduction to PYNQ framework

ADIUVO
ENGINEERING AND TRAINING, LTD.

What is PYNQ?

PYNQ is an open source project started by X
productivity of Python with the acceleration pl
within the Zynq / Zynq MPSoC

Hosted at PYNQ.io

Language Rank	Types	Spectrum Ranking
1. Python		100.0
2. C++		99.7
3. Java		97.5
4. C		96.7
5. C#		89.4
6. PHP		84.9
7. R		82.9
8. JavaScript		82.6
9. Go		76.4
10. Assembly		74.1

Why should I learn PYNQ ?

Tight coupling of processing system (PS) and programmable logic (PL) in the Zynq / Zynq MPSoC creates a system which is

- Responsive – Leverage the parallel processing capability provided by the PL
- Deterministic – Creates processing pipeline competing for fewer shared resources
- Power Efficient – Less off chip transactions to or from DDR memory, dedicated hardware implementation is more efficient than

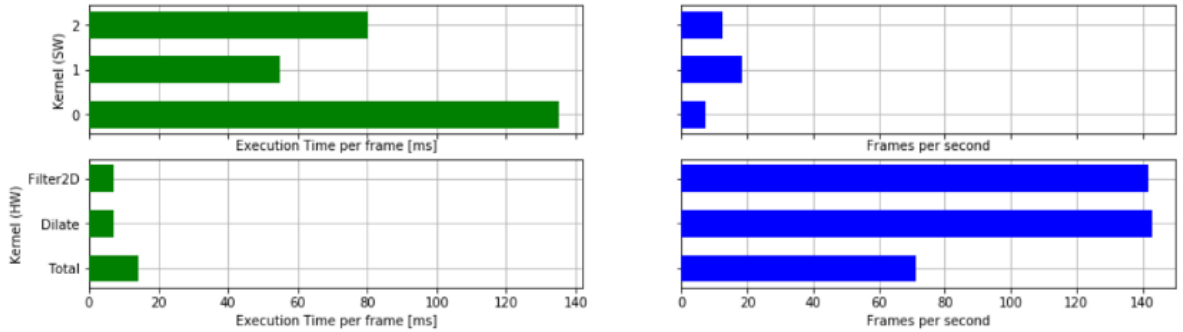
PYNQ frees Python programmers from the sequential software world and opens up the acceleration of programmable logic without the need to be a digital designer.

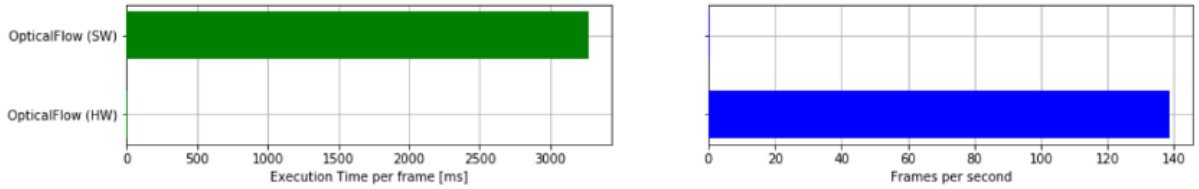
Why should I learn PYNQ ?

Simple example of PYNQ in a image processing application

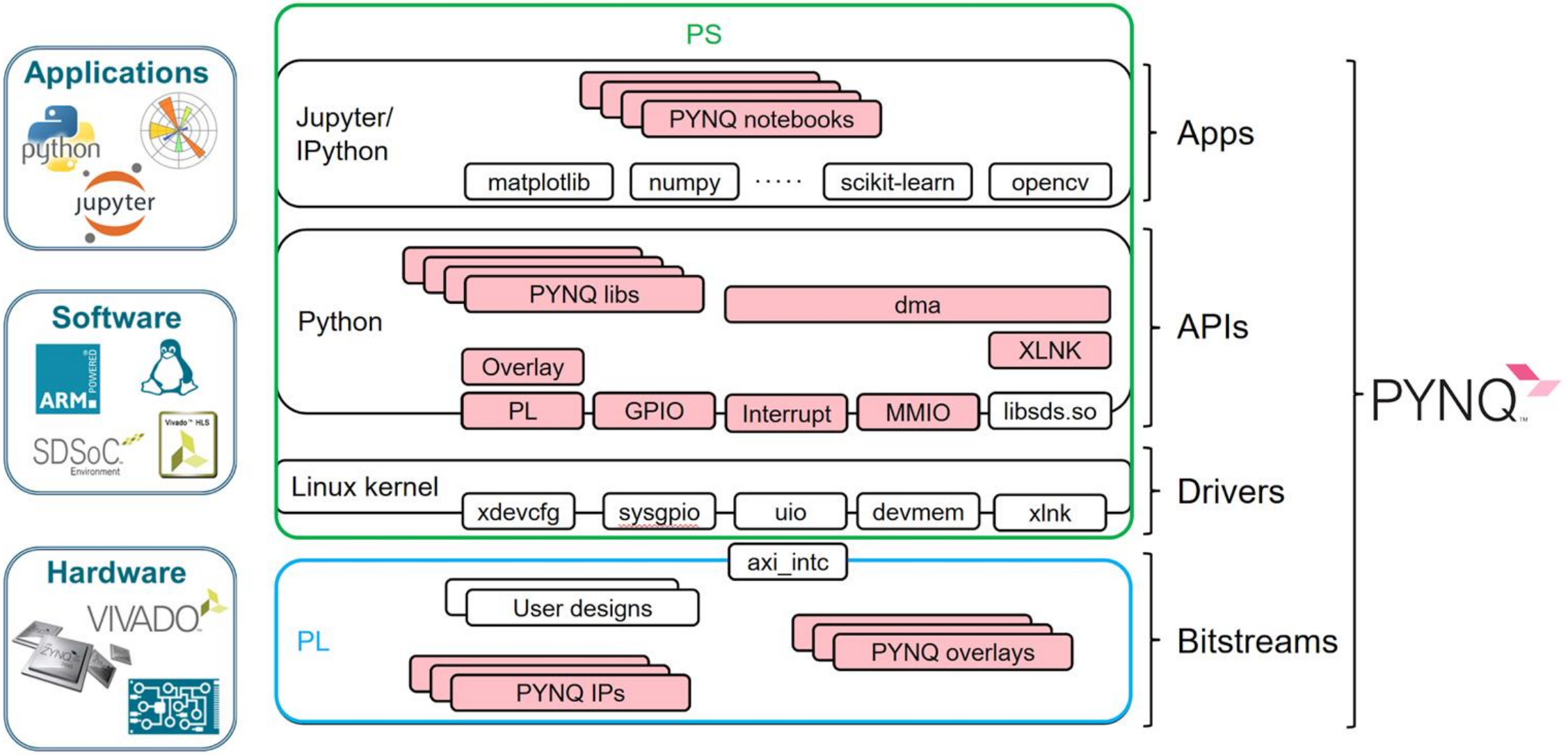
- Image Filtering
  - SW < 20 Frames per Second
  - HW > 60 Frames per Second

- Optical Flow
  - SW < 1 Frame per Second
  - HW > 120 Frames per Second





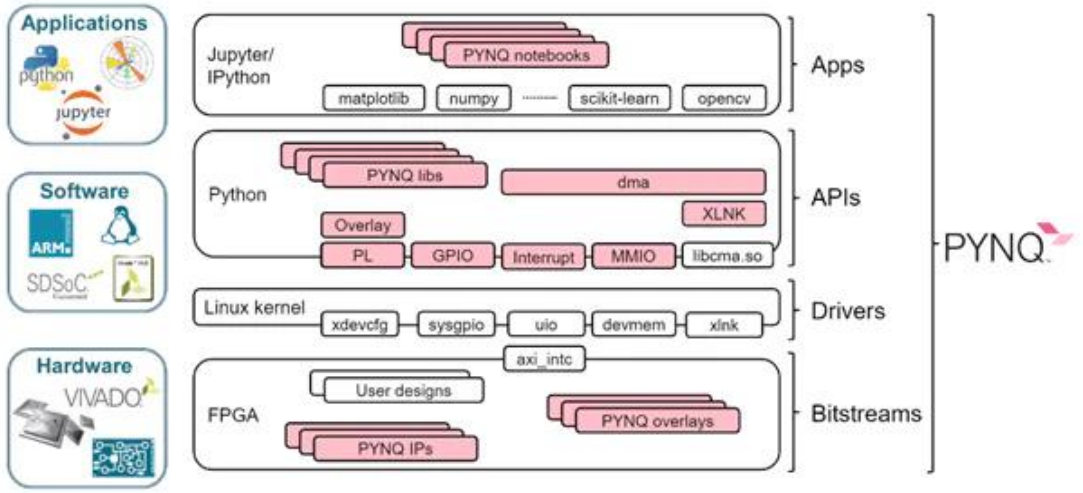
PYNQ Framework: Interfacing Python with Xilinx SoC



PYNQ Components

To achieve performance PYNQ is comprised of

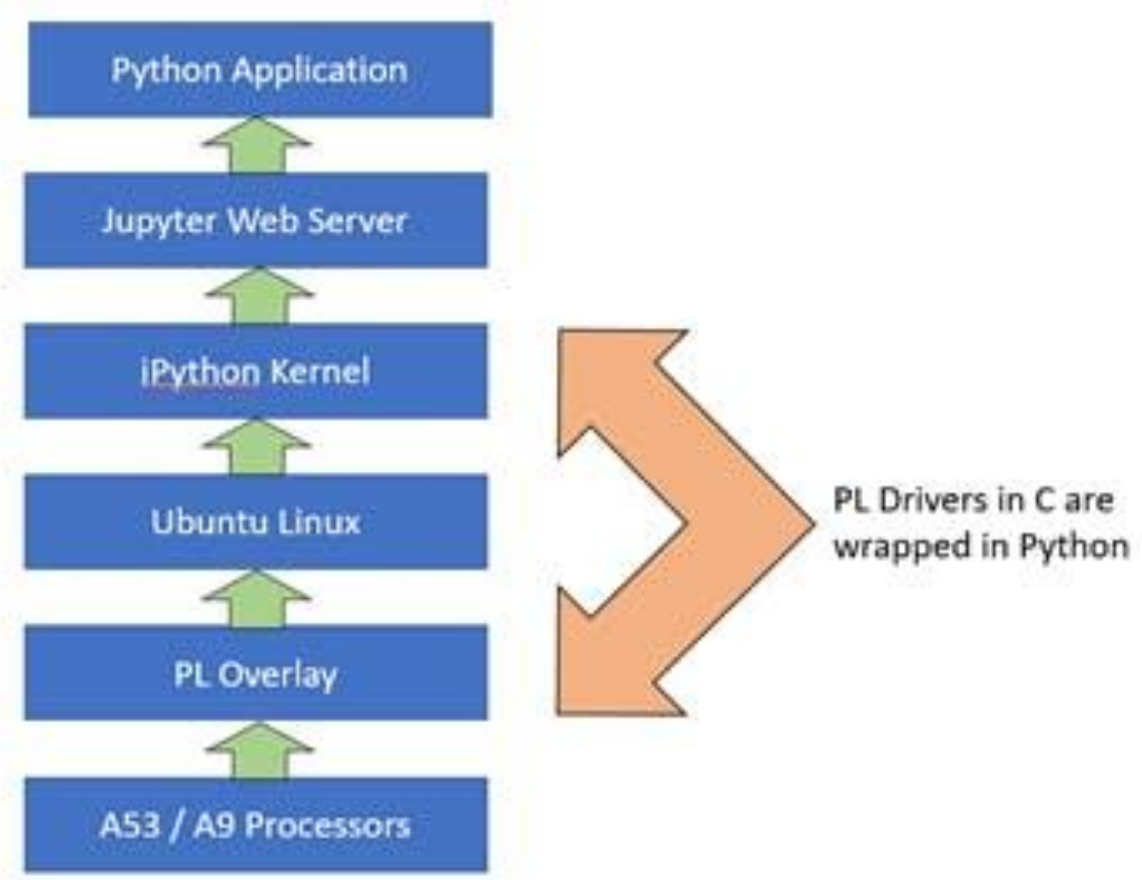
- Jupyter Notebooks
- PYNQ Package
- PYNQ Libs
- PYNQ Classes
- PYNQ IP
- PYNQ Overlays



PYNQ Architecture

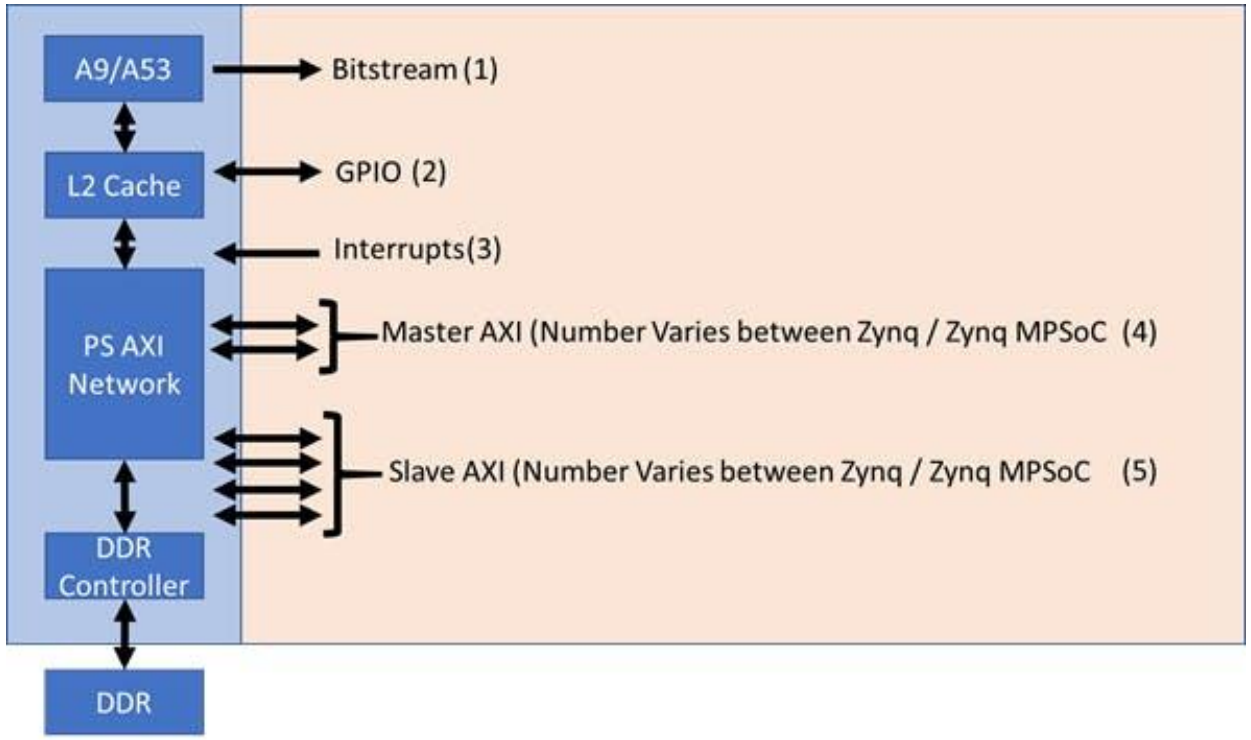
Pynq is built upon Xilinx Petalinux flow

Standard way to create PYNQ for a custo
we will see



Working with the PL

1. Bitstream — This configures the programmable logic for the desired application. In the PYNQ framework, the **xdevcfg** driver is used.
2. GPIO — This provides simple IO in both directions. In the PYNQ framework, this is supported by the **sysgpio** driver.
3. Interrupts — Support interrupt generation from the programmable logic to the processing system. In the PYNQ framework, this is supported by the **Userspace IO** driver.
4. Master AXI Interfaces — These are used to transfer data between the PS to the PL when the PS is the initiator of the transaction. The PYNQ framework uses **devmem** when employing master AXI interface.
5. Slave AXI Interfaces — These are used to transfer data between the PS and PL when the PL is the initiator of the transaction. The PYNQ framework uses **xlnk** to enable these transfers.



Available PYNQ boards

Several PYNQ builds for existing boards:

- Pynq Z1 — Zynq SoC 7020
- Pynq Z2 — Zynq SoC 7020
- ZCU104 — Zynq MPSoC XCZU7EV
- ZCU111 — Zynq RFSoC XCZU28DR
- Ultra96 — Zynq MPSoC ZU3EG

Arty Z7-20 can use the Pynq Z1 image a basically same board.

Overlays

Overlays are the design loaded into the programmable logic

Can be custom created or accessed via the PYNQ.IO community

Range of Overlays in the community including

- Machine Learning
- Image Processing
- RISC-V
- Kalman filter

Base Overlay is the initial overlay which is created with the PYNQ Image

Download and work with new overlays as required

Of course you can also create you own – As we will see

PYNQ Libraries

PYNQ provides several libraries which provide support for management of the processor and allow access to the low level hardware including

IP Cores – Audio, AXI GPIO, AXI IIC, DMA, Logic Tools, Video

IOP – Arduino, Grove, RPI, PMOD

PYNQ MicroBlaze – MicroBlaze Subsystem RPC and Library

PS / PL Interface – Interrupt, MMIO, PS GPIO, Xlnk

PS Control – PMBus

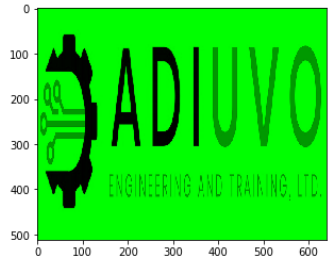
PL Control – Overlay, PL and Bitstream Classes

PYNQ In Industry

Working with THALES – Major Defense Client – High Speed Image processing (3000 > 10,000 FPS)

ZCU102 Development board

“The use of Pynq to rapidly test and evaluate design patterns for image processing has been invaluable. Not only has it sped up design, but it also reduces the necessary team size to a manageable level.”




```
In [21]: from pynq.ps import Clocks
         Clocks.fclk0_mhz = 350
         Clocks.fclk0_mhz

         /usr/local/lib/python3.6/dist-packages/pynq/ps.py:312: UserWarning: Setting frequency to the closet possible value 374.99625MHz.
           round(freq_high / q0, 5)))
Out[21]: 374.99625

In [22]: gpio = overlay.axi_gpio_0
         gpio.write(0x00,374996250)

In [33]: FPS = gpio.read(0x08)
         print("frames per second =",FPS)

         frames per second = 1121
```

Accelerating FFT with PYNQ

ADIUVO
ENGINEERING AND TRAINING, LTD.

Lab : FFT Acceleration

Open a browser and go to

www.pynq.io

Lab : FFT Acceleration

Select the boards page and

download the SD card

image for Pynq Z1 v2.6

Home | Get Started | Boards | Community | Source Code | Support

Development Boards

PYNQ supports Zynq based boards (Zynq, Zynq Ultrascale+, Zynq RFSoC), and **Xilinx Alveo** accelerator boards and **AWS-F1** instances.
See the **PYNQ Alveo Getting Started guide** for details on installing PYNQ for use with Alveo and AWS-F1.

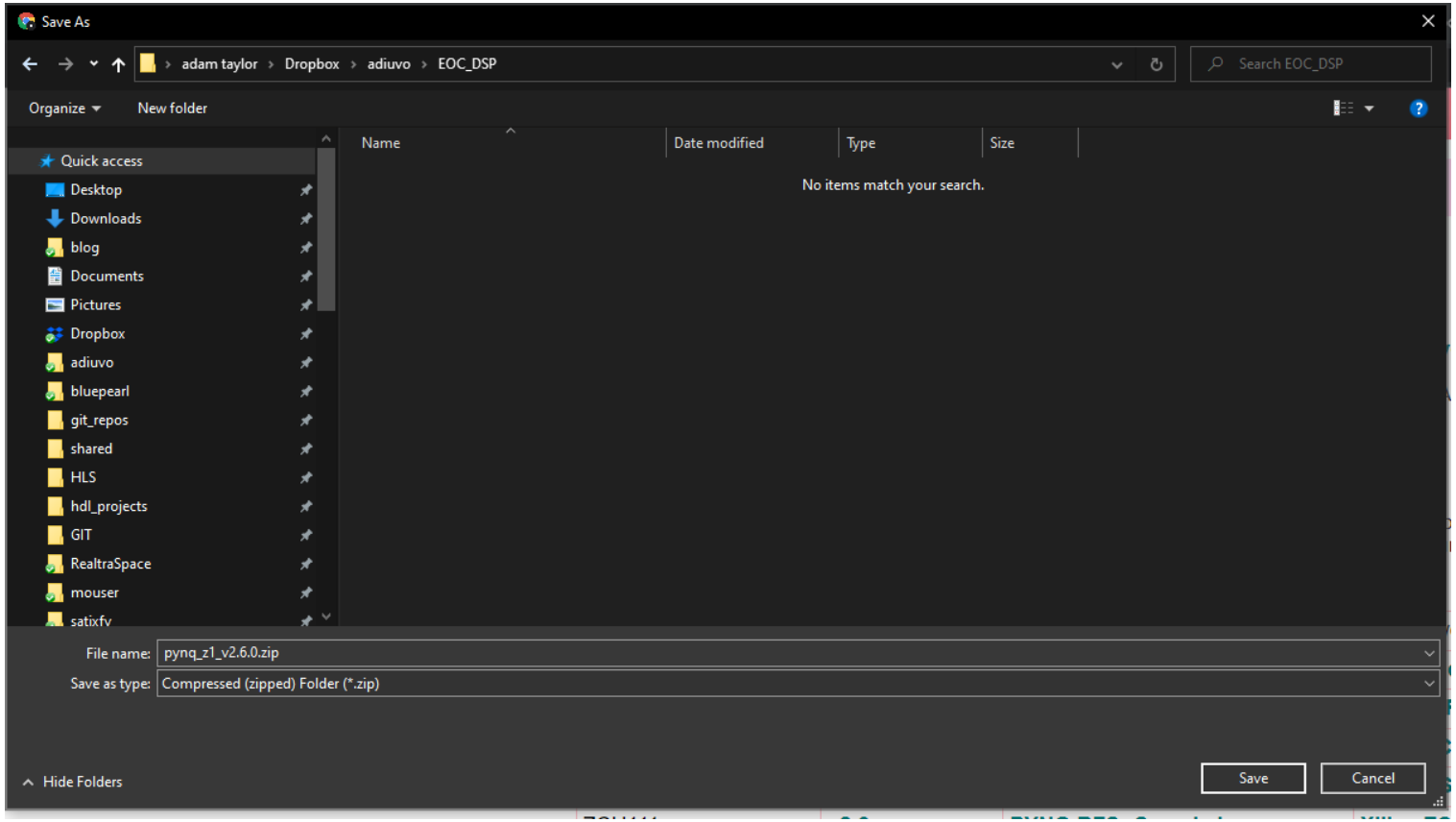
Downloadable PYNQ images

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	Documentation	Vendor webpage
PYNQ-Z2	v2.6	PYNQ setup guide	TUL Pynq-Z2
PYNQ-Z1	v2.6	PYNQ setup guide	Digilent Pynq-Z1
ZCU104	v2.6	PYNQ setup guide	Xilinx ZCU104
RFSoC 2x2	v2.6	RFSoC 2x2 GitHub Pages	XUP RFSoC 2x2
ZCU111	v2.6	PYNQ RFSoC workshop	Xilinx ZCU111
Ultra96V2	v2.6	Avnet PYNQ documentation	Avnet Ultra96V2
Ultra96 (legacy)	v2.6	See Ultra96V2	See Ultra96V2
TySOM-3-ZU7EV	v2.5	GitHub project page	Aldec TySOM-3-ZU7EV
TySOM-3A-ZU19EG	v2.5	GitHub project page	Aldec TySOM-3A-ZU19EG

Lab : FFT Acceleration

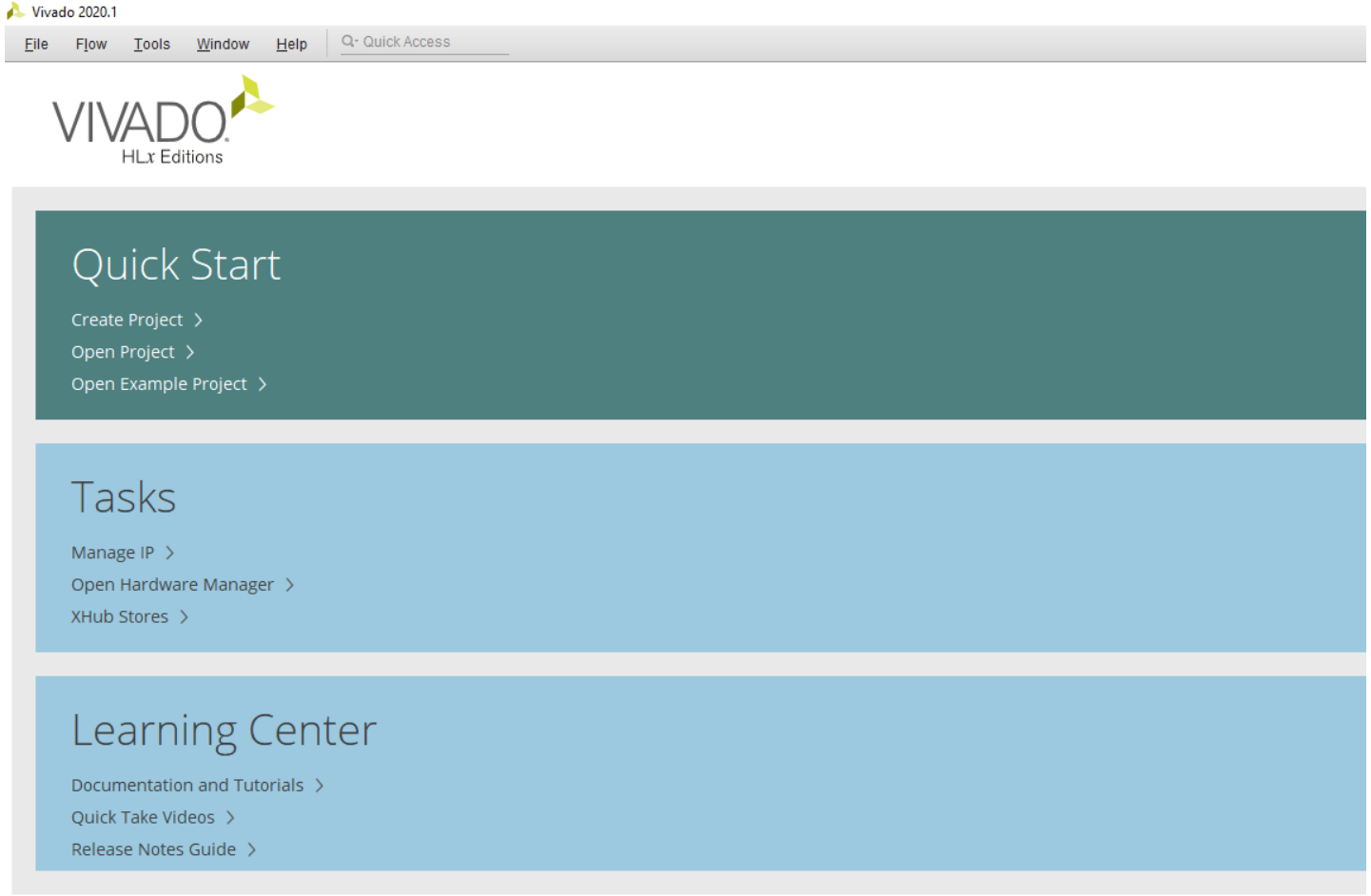
Save the SD Card image to a preferred location on your local computer



Lab : FFT Acceleration

Open Vivado and select Xhub

Stores



Lab : FFT Acceleration

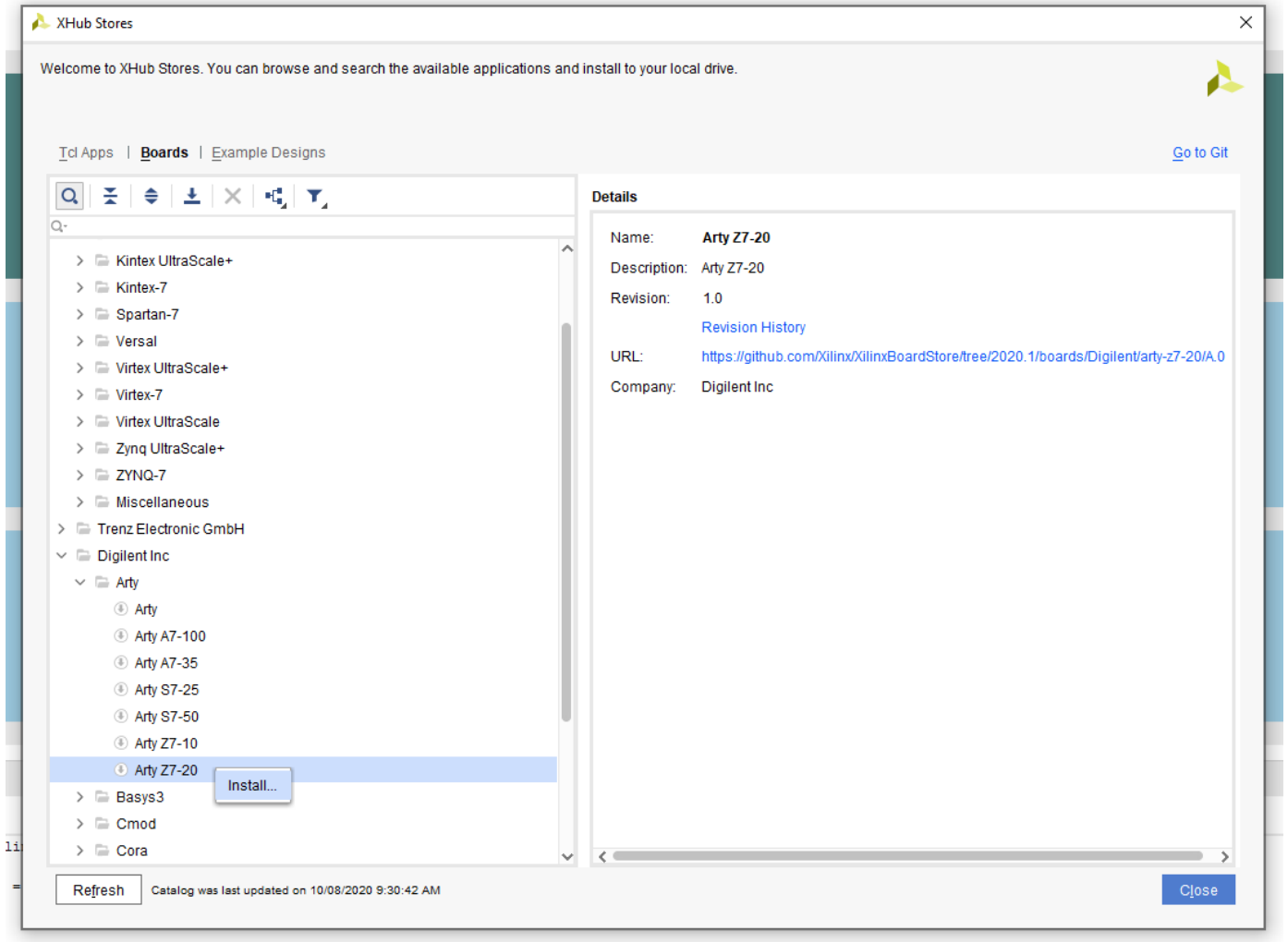
From the boards tab, select

Digilent Inc folder. Expand the

Arty directory and select Arty.

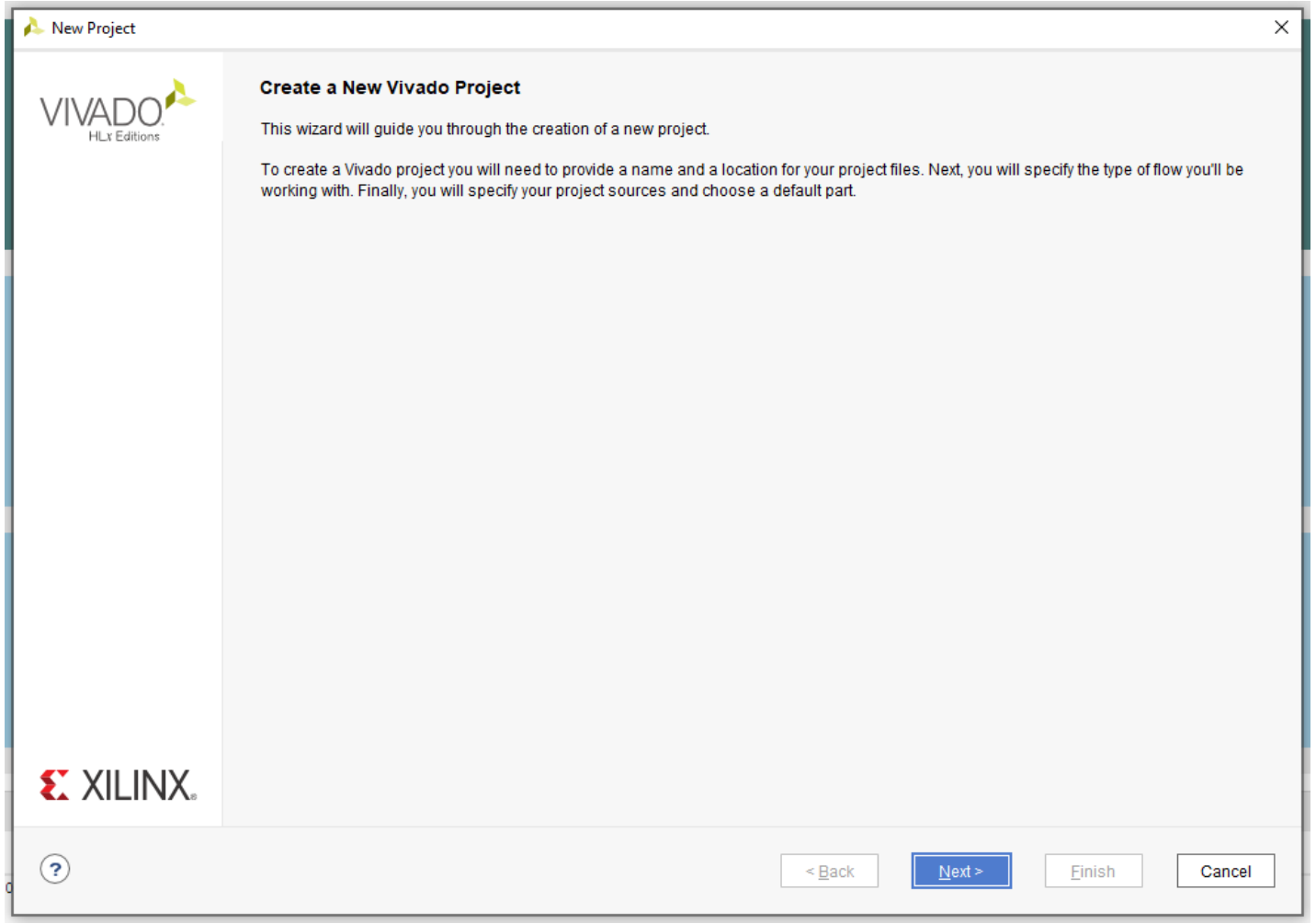
Right click on the Arty Z7-20

and select install



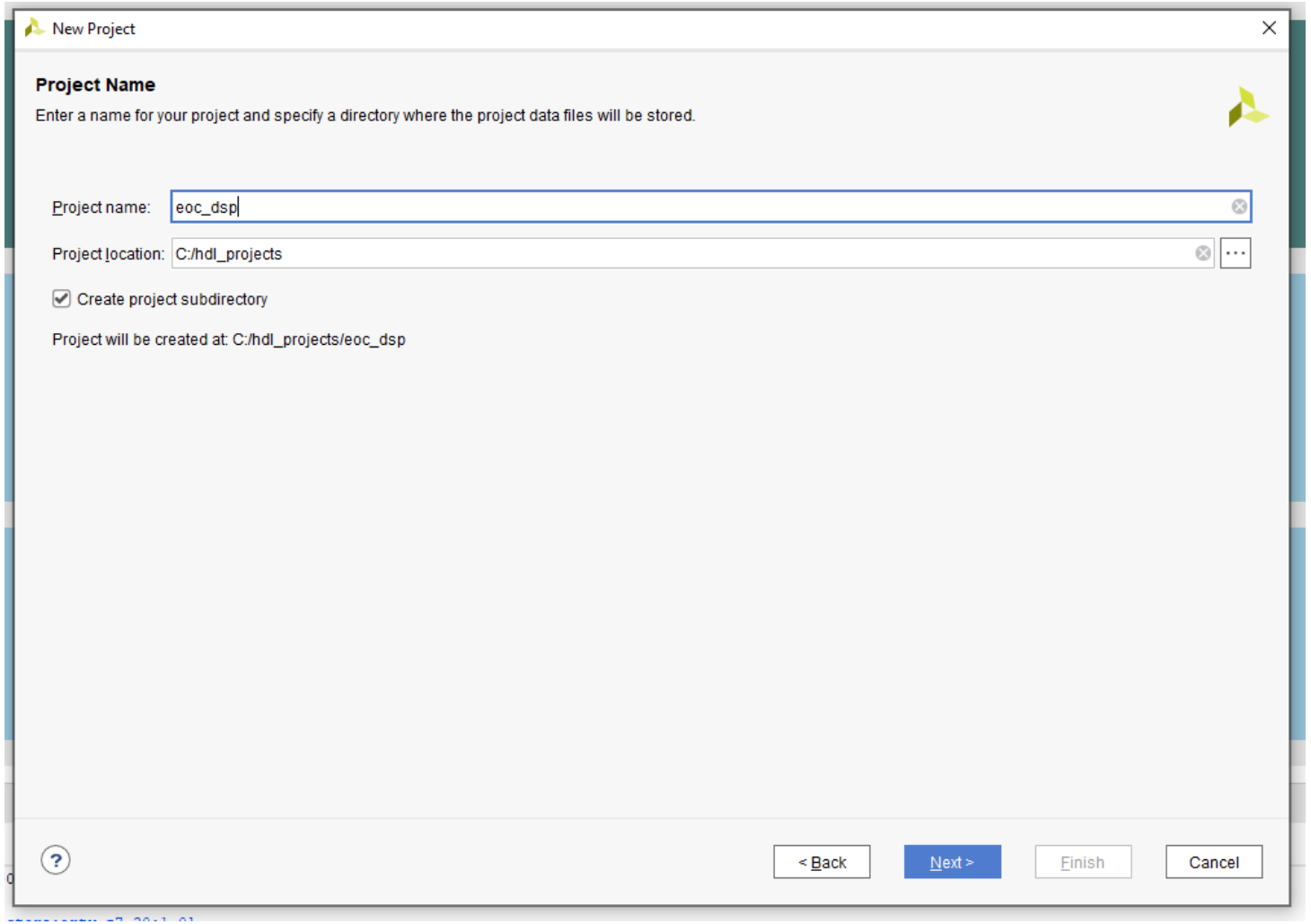
Lab : FFT Acceleration

Create a new project



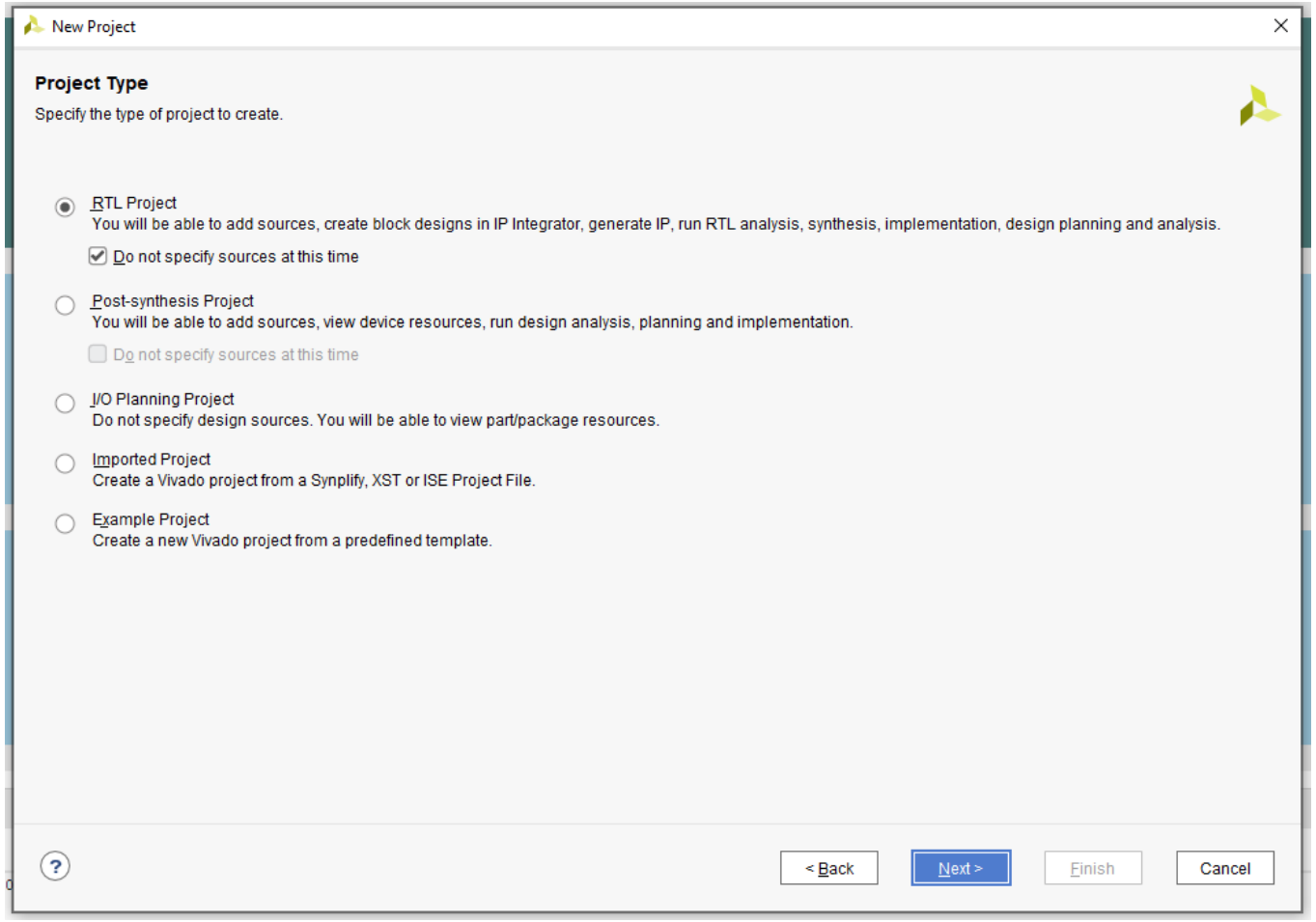
Lab : FFT Acceleration

Enter a name and location



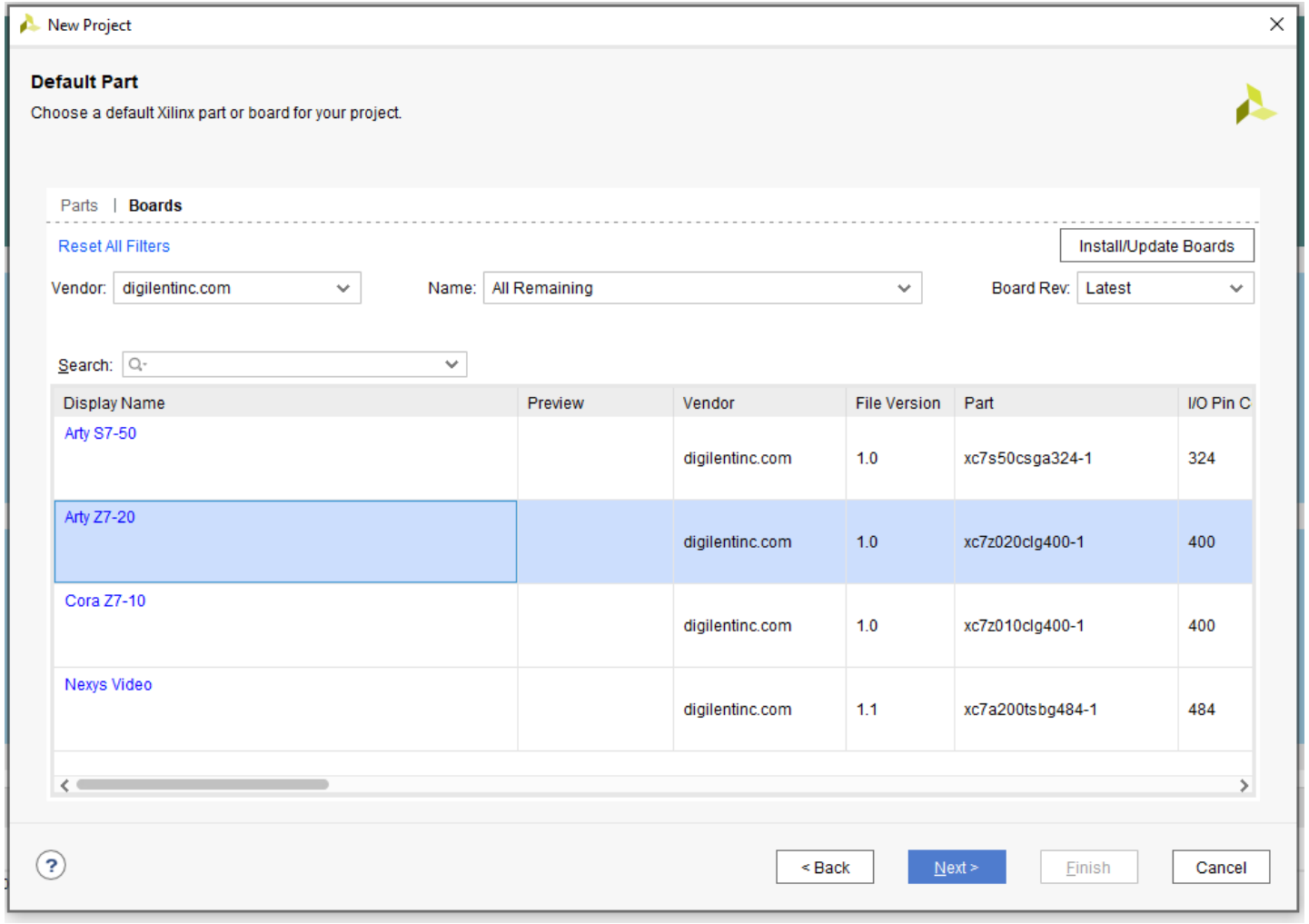
Lab : FFT Acceleration

Select RTL Project



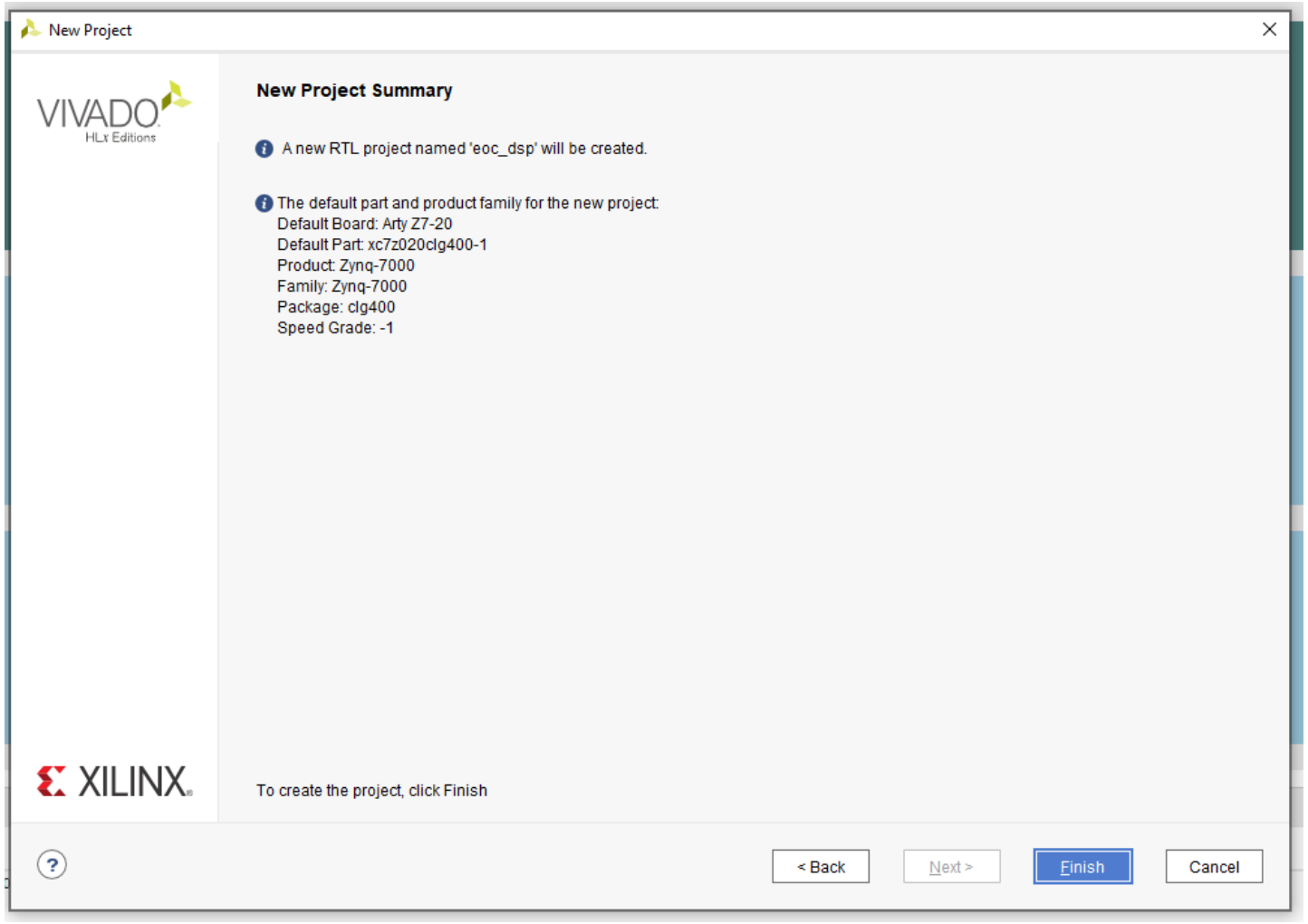
Lab : FFT Acceleration

Select the Arty Z7-20 board



Lab : FFT Acceleration

Click Finish to create the project

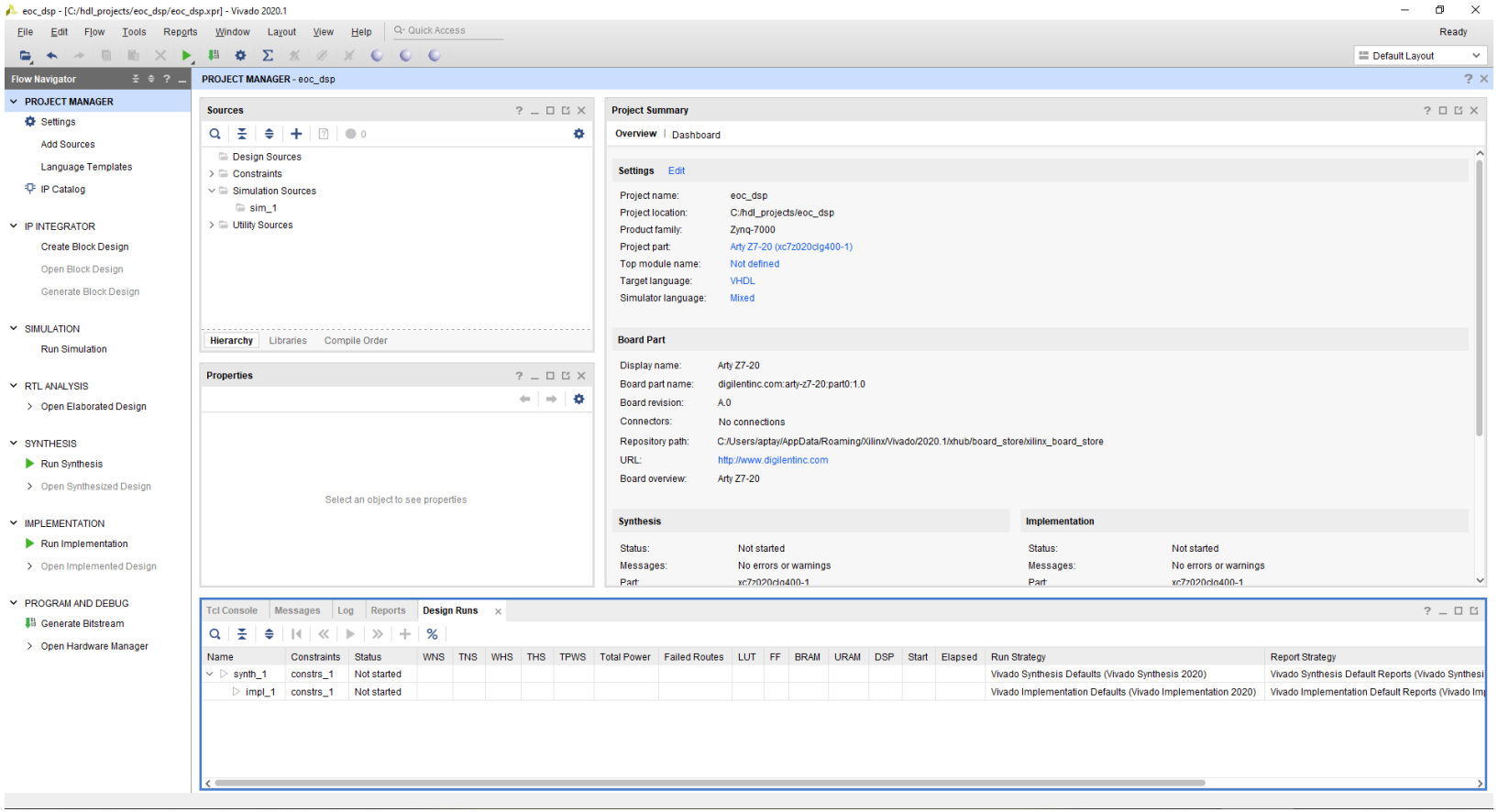


Lab : FFT Acceleration

From the Project

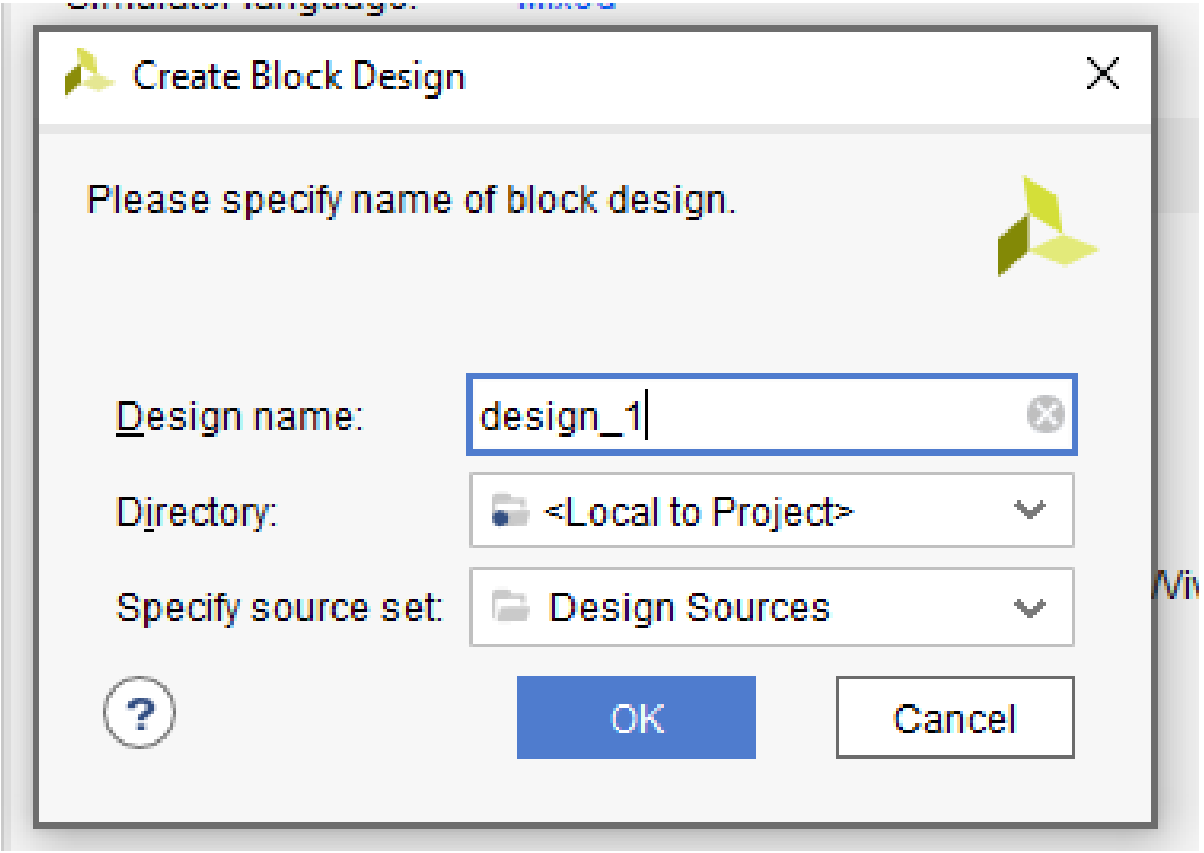
Manager, select create

block diagram



Lab : FFT Acceleration

Leave, defaults unchanged and click OK

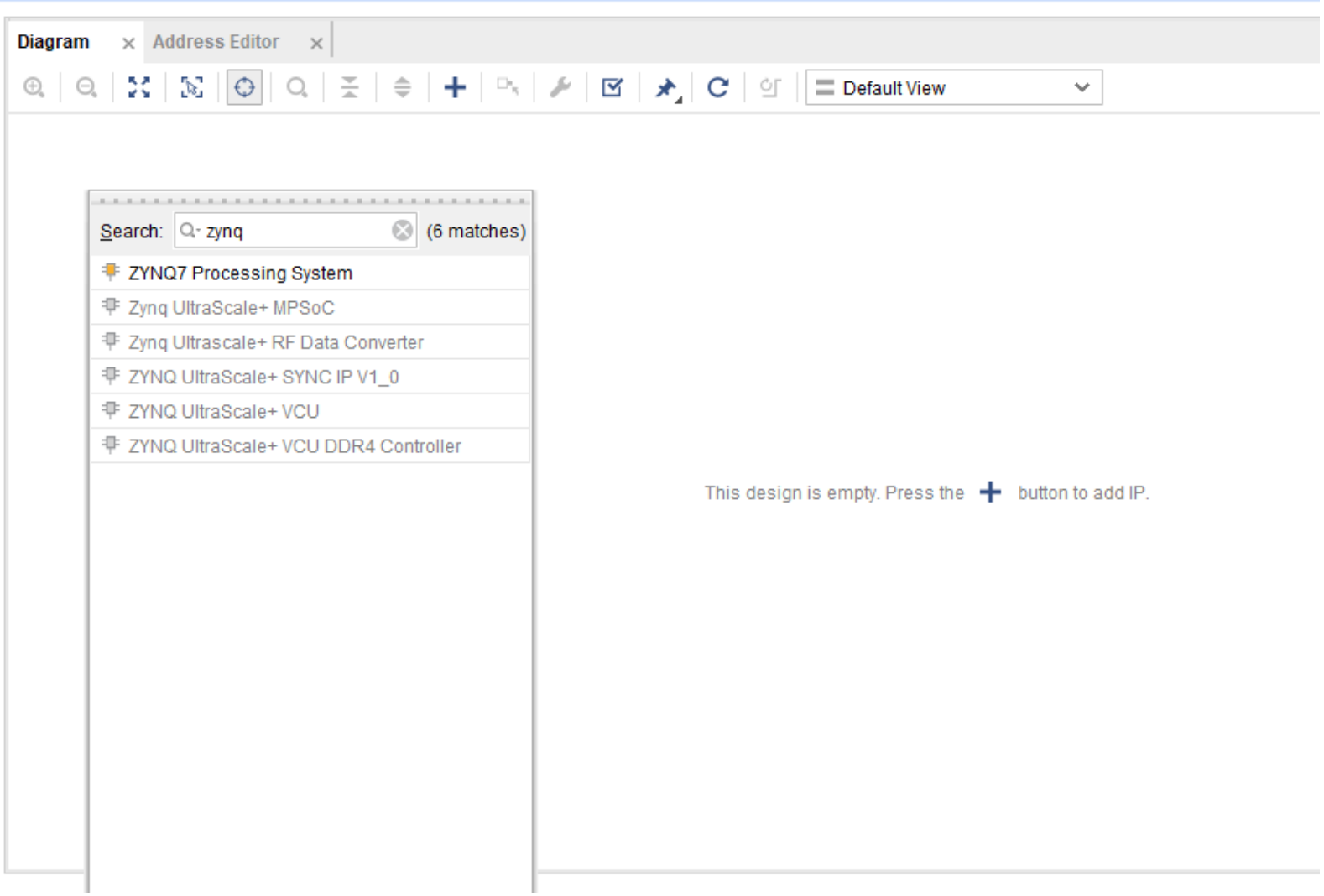


Lab : FFT Acceleration

Click on + and in the

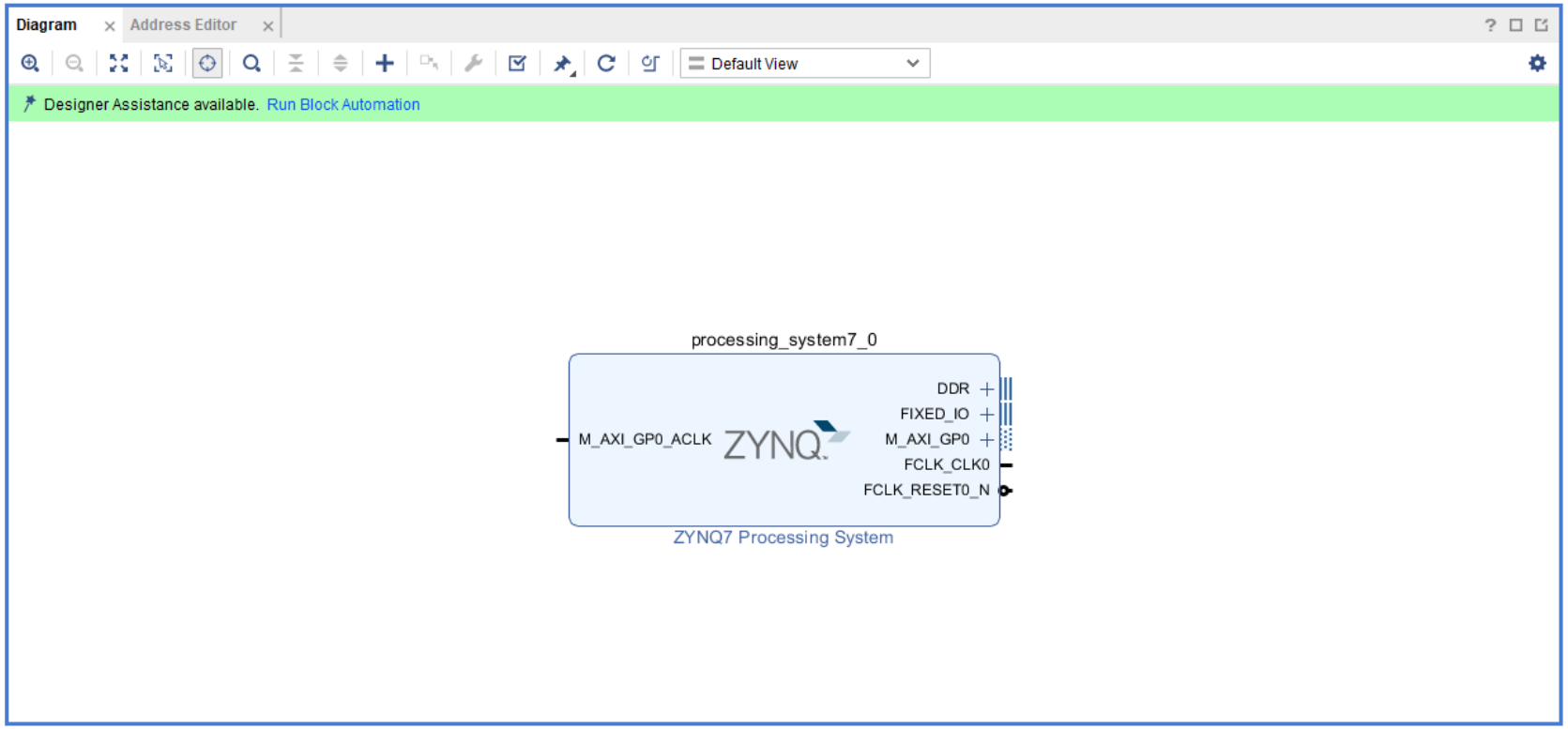
search bar type in Zynq

and press enter



Lab : FFT Acceleration

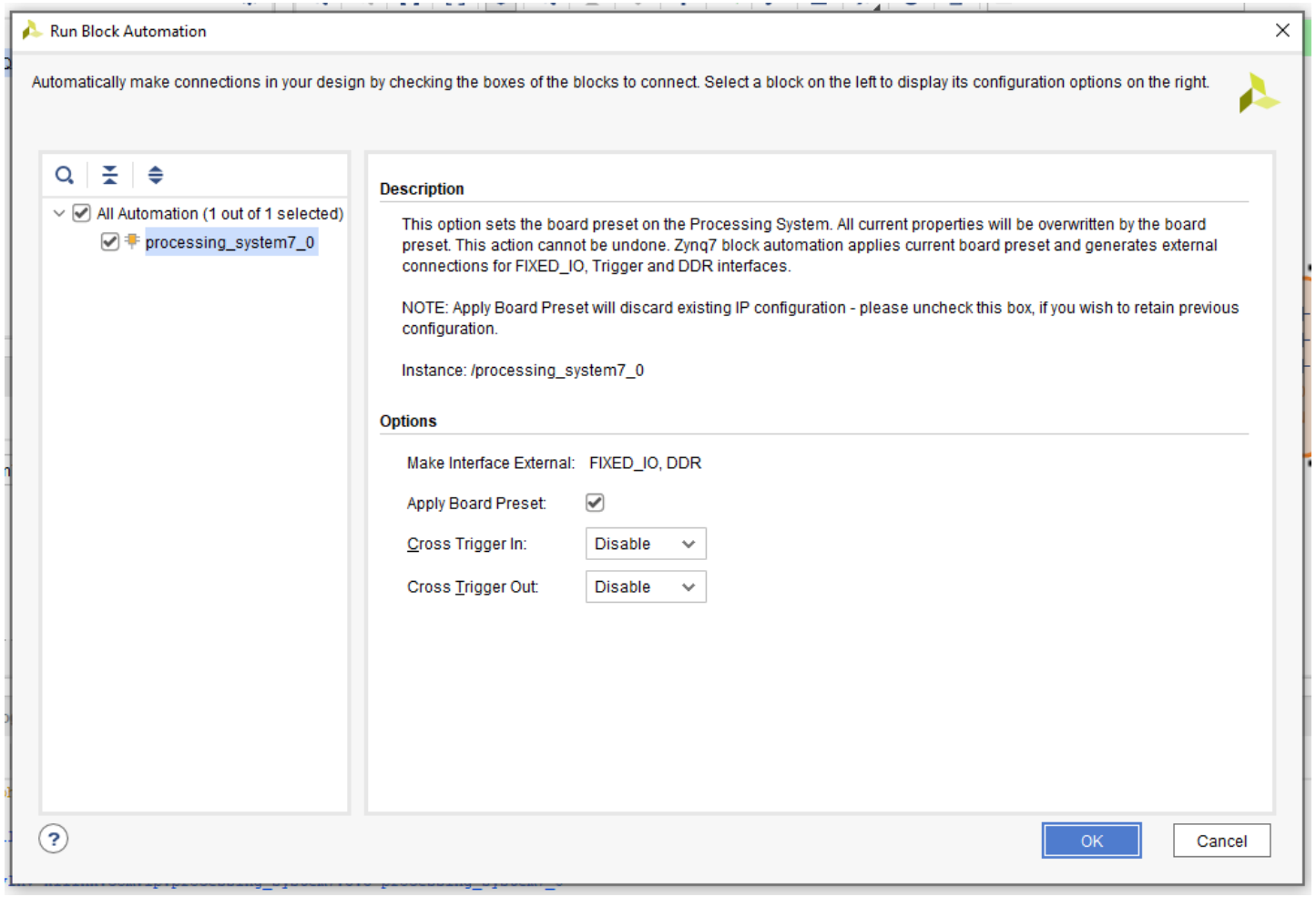
Run the block automation



Lab : FFT Acceleration

Leave the settings as default and

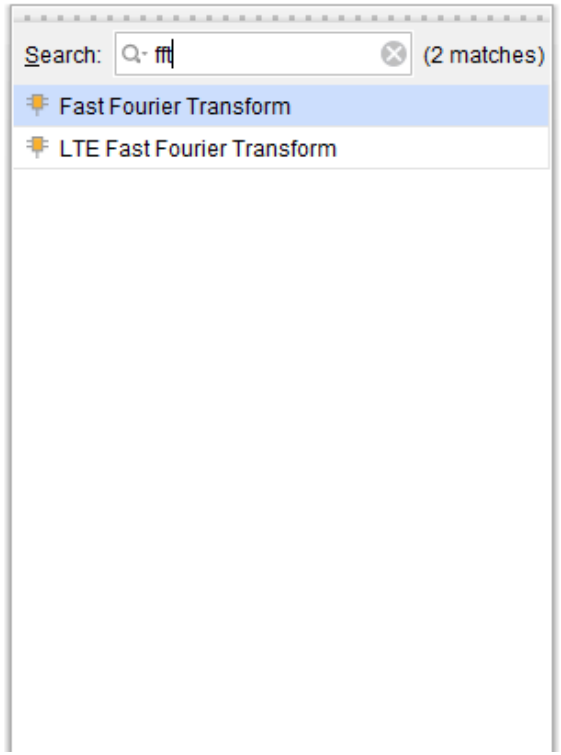
click OK

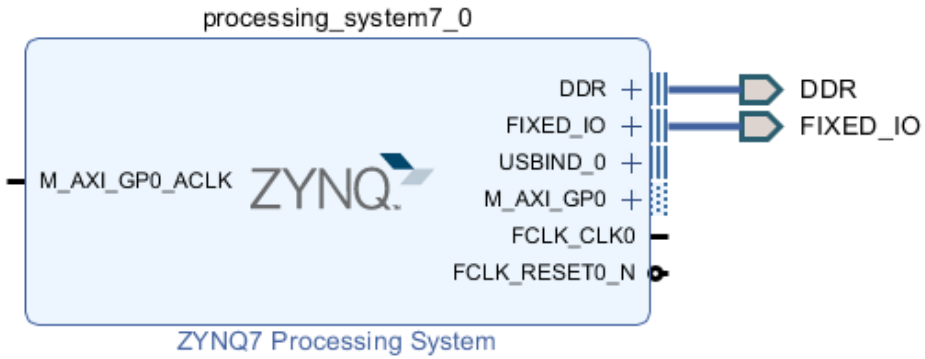


Lab : FFT Acceleration

Click on + and add in the

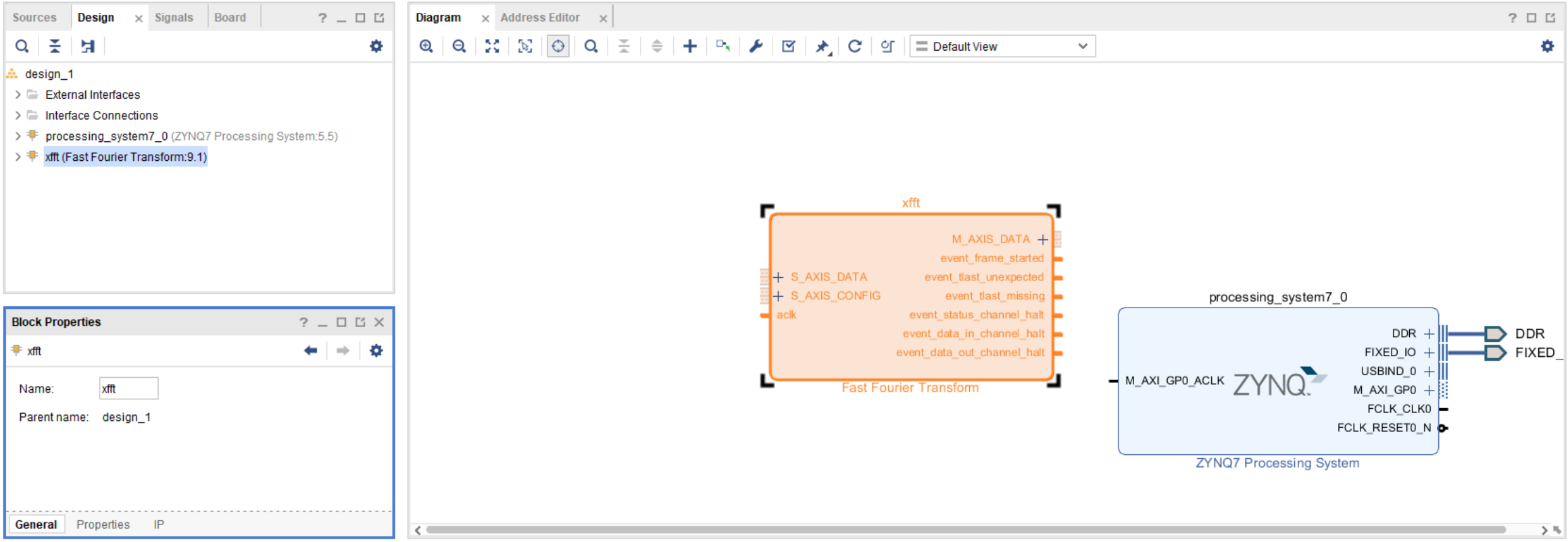
FFT





Lab : FFT Acceleration

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

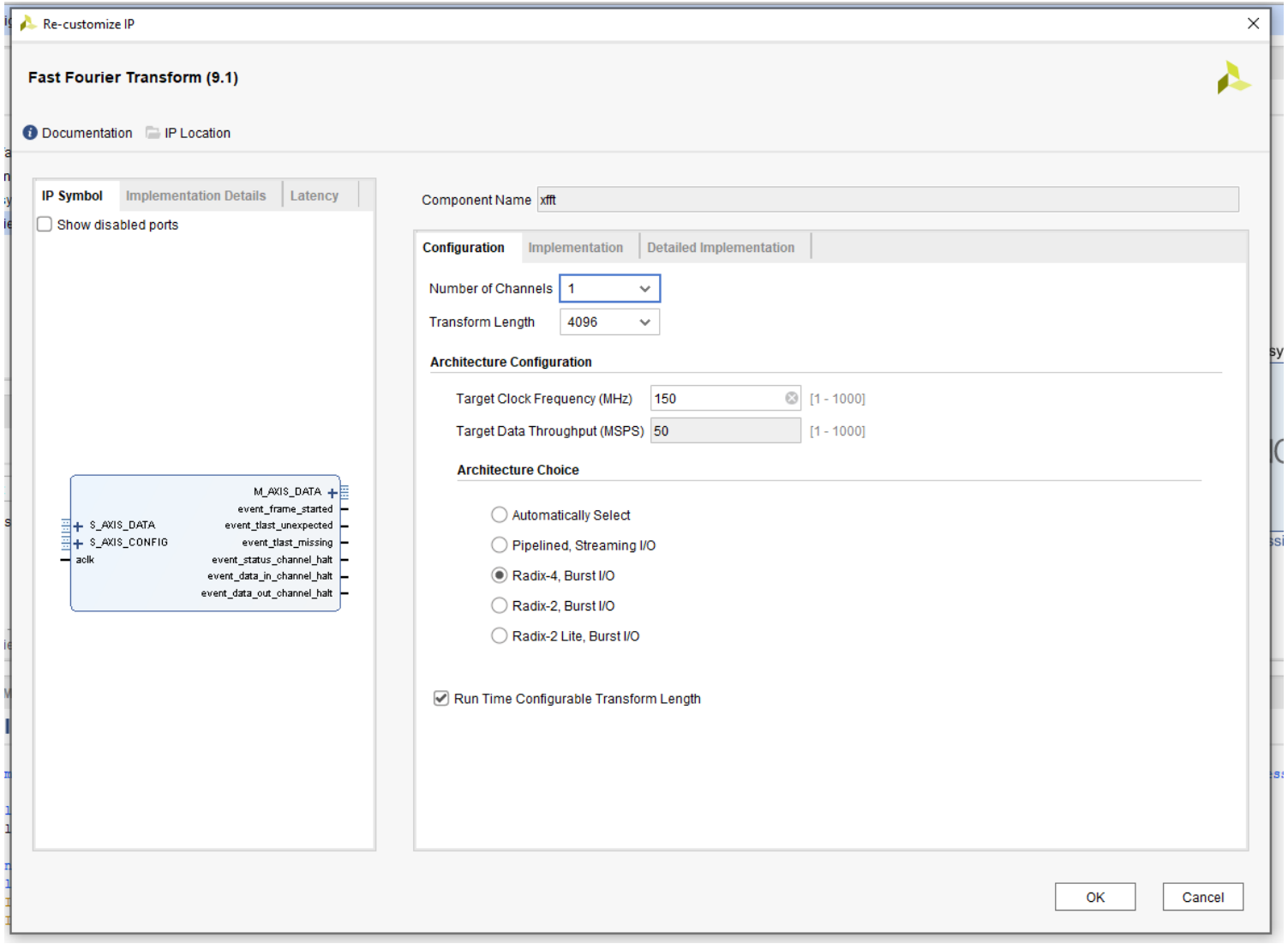


Lab : FFT Acceleration

On the configuration tab, select

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

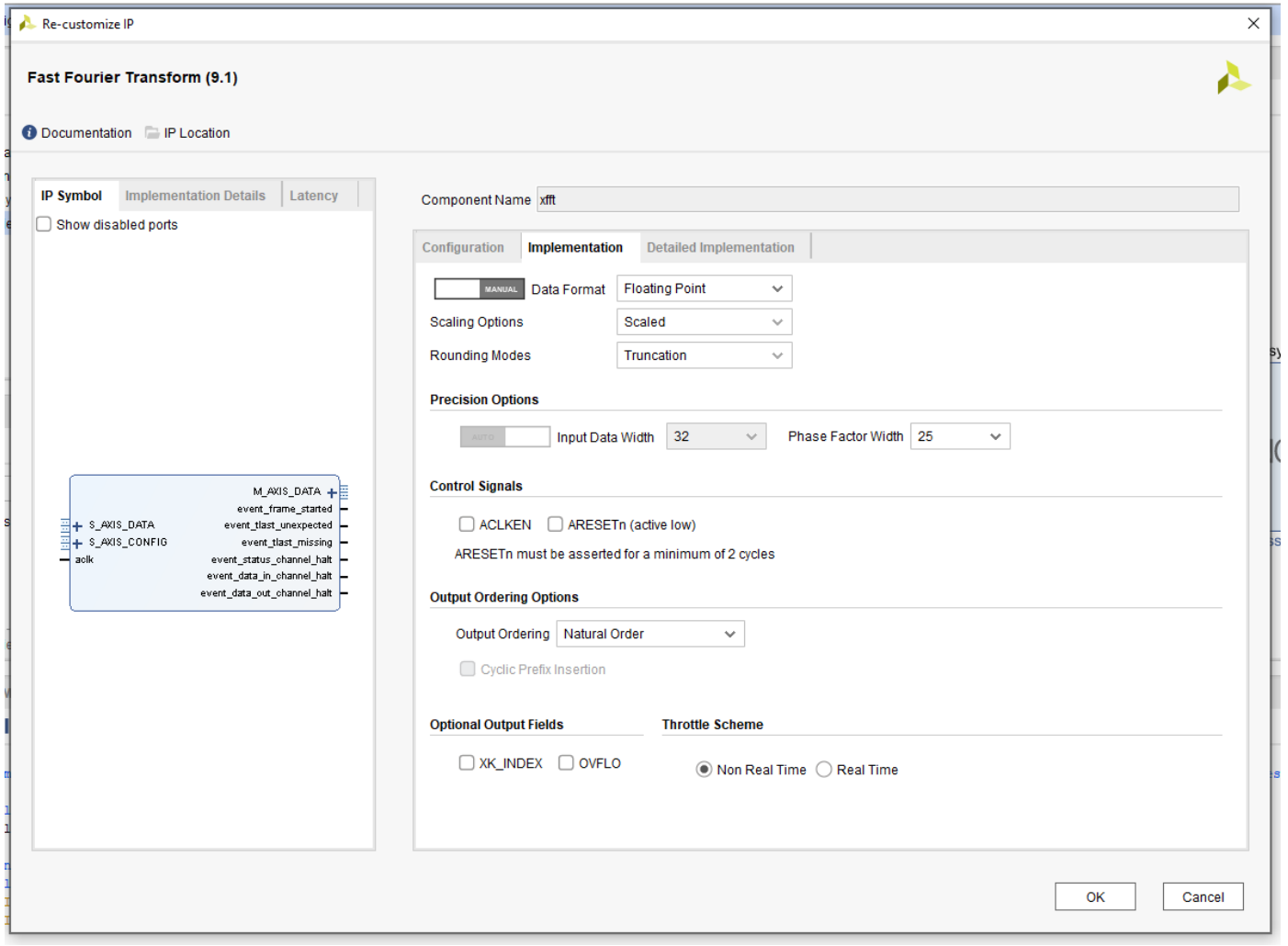
  Configurable transform length



Lab : FFT Acceleration

On the implementation tab select

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

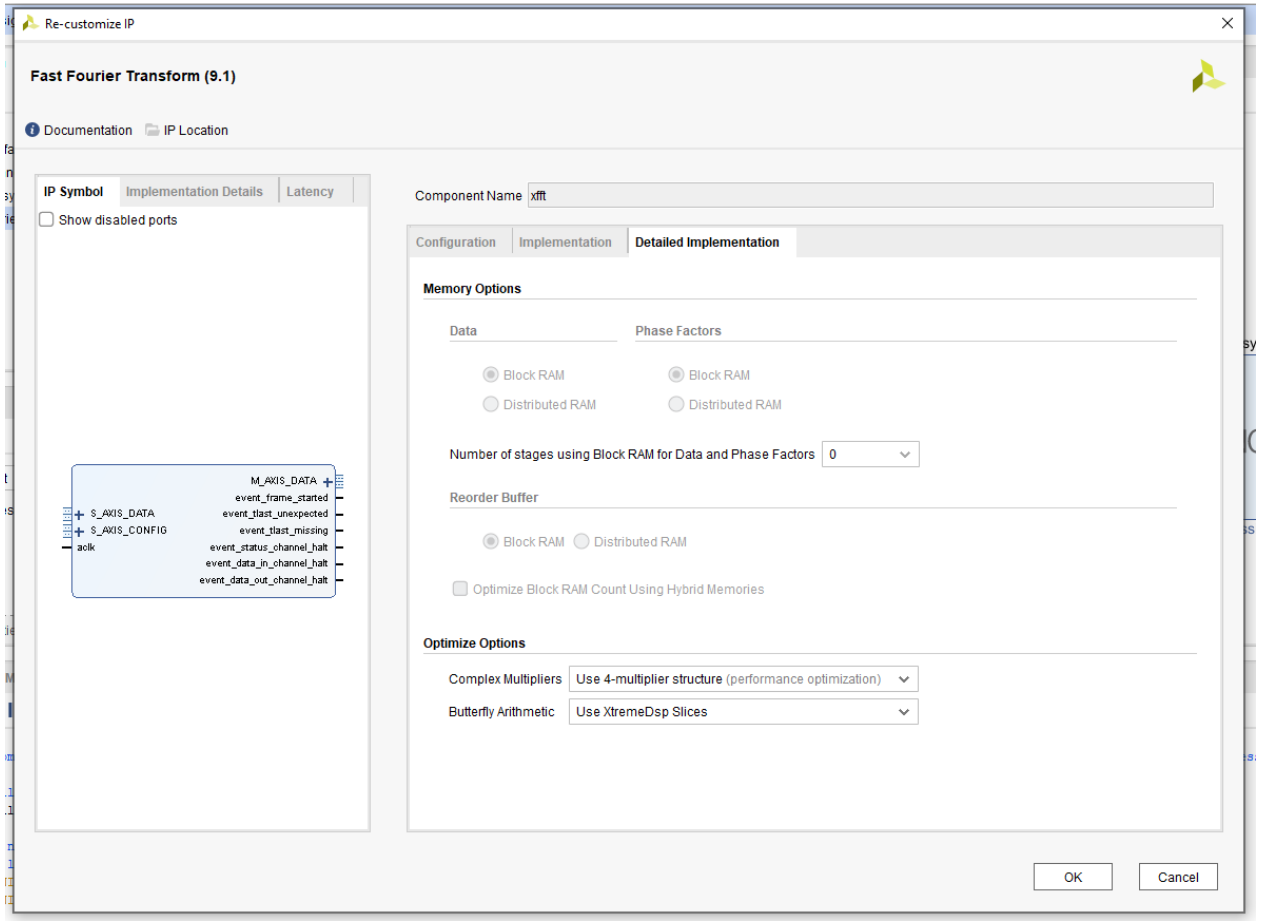


Lab : FFT Acceleration

On the Detailed Implementation tab

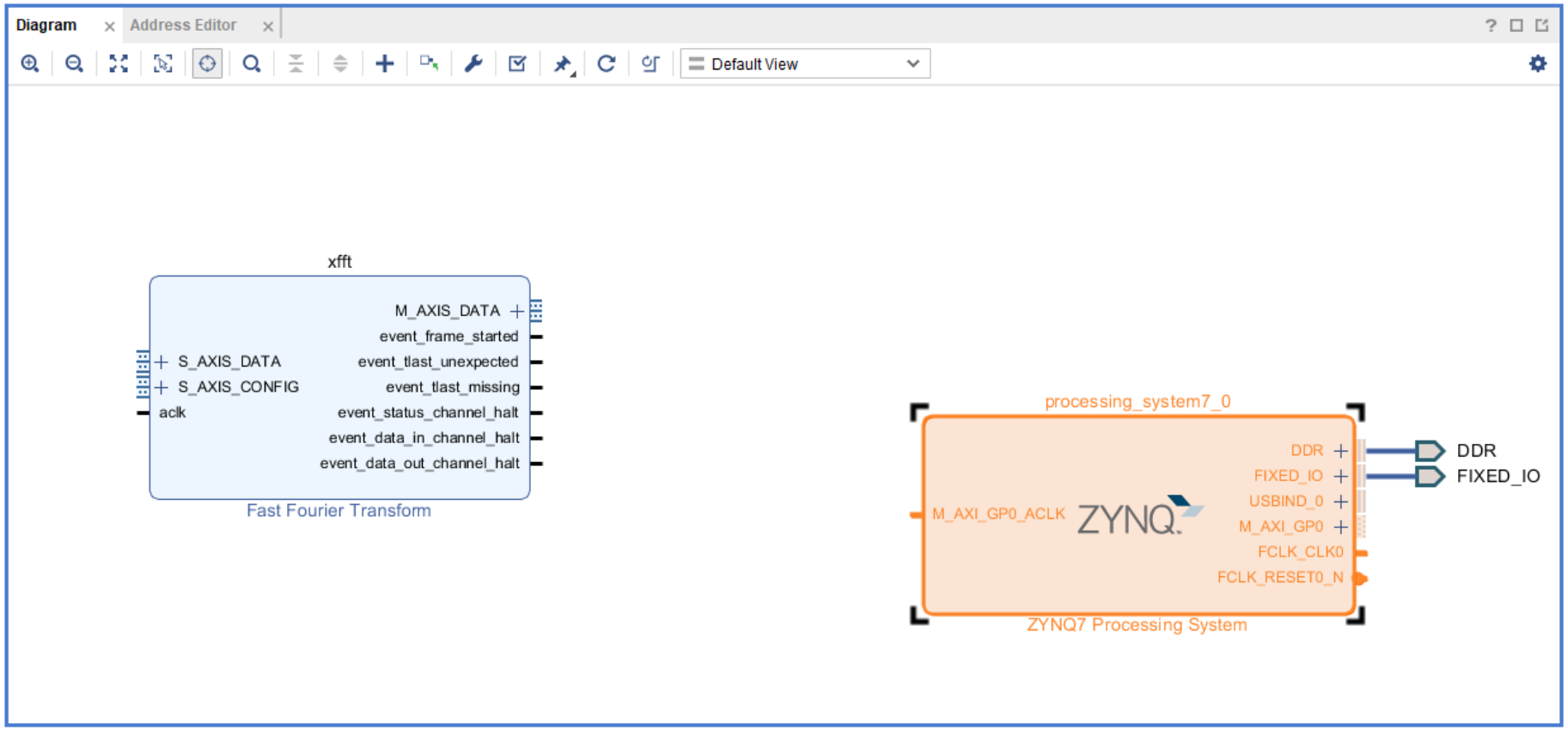
select

- Use 4-Multipler Structure
- Use XtremeDSP Slices



Lab : FFT Acceleration

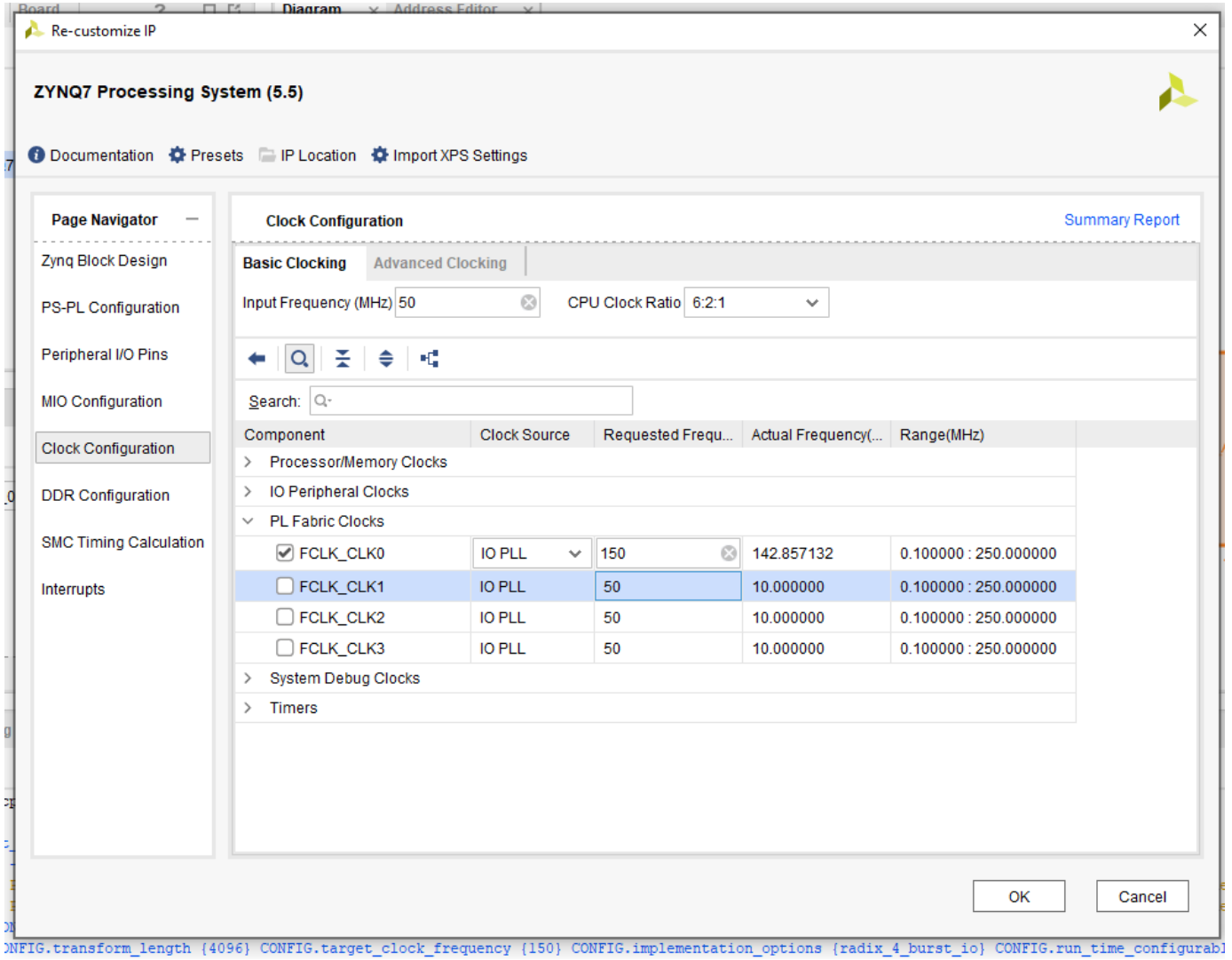
Double click on the Processing System to reconfigure it



Lab : FFT Acceleration

On the clocking tab change the

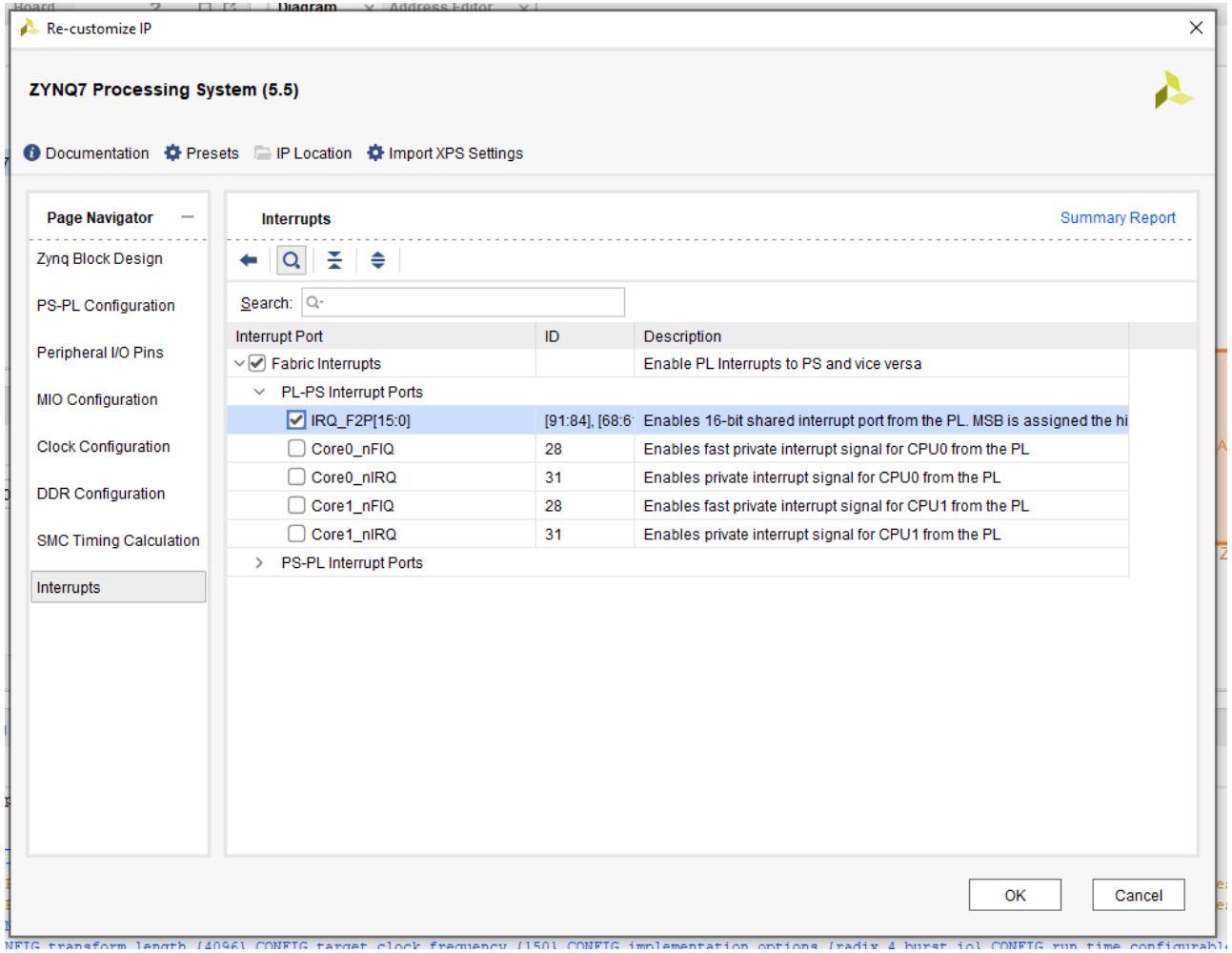
frequency of clock one to 150MHz



Lab : FFT Acceleration

On the Interrupts Tab enable the

IRQ_F2P[15:0]

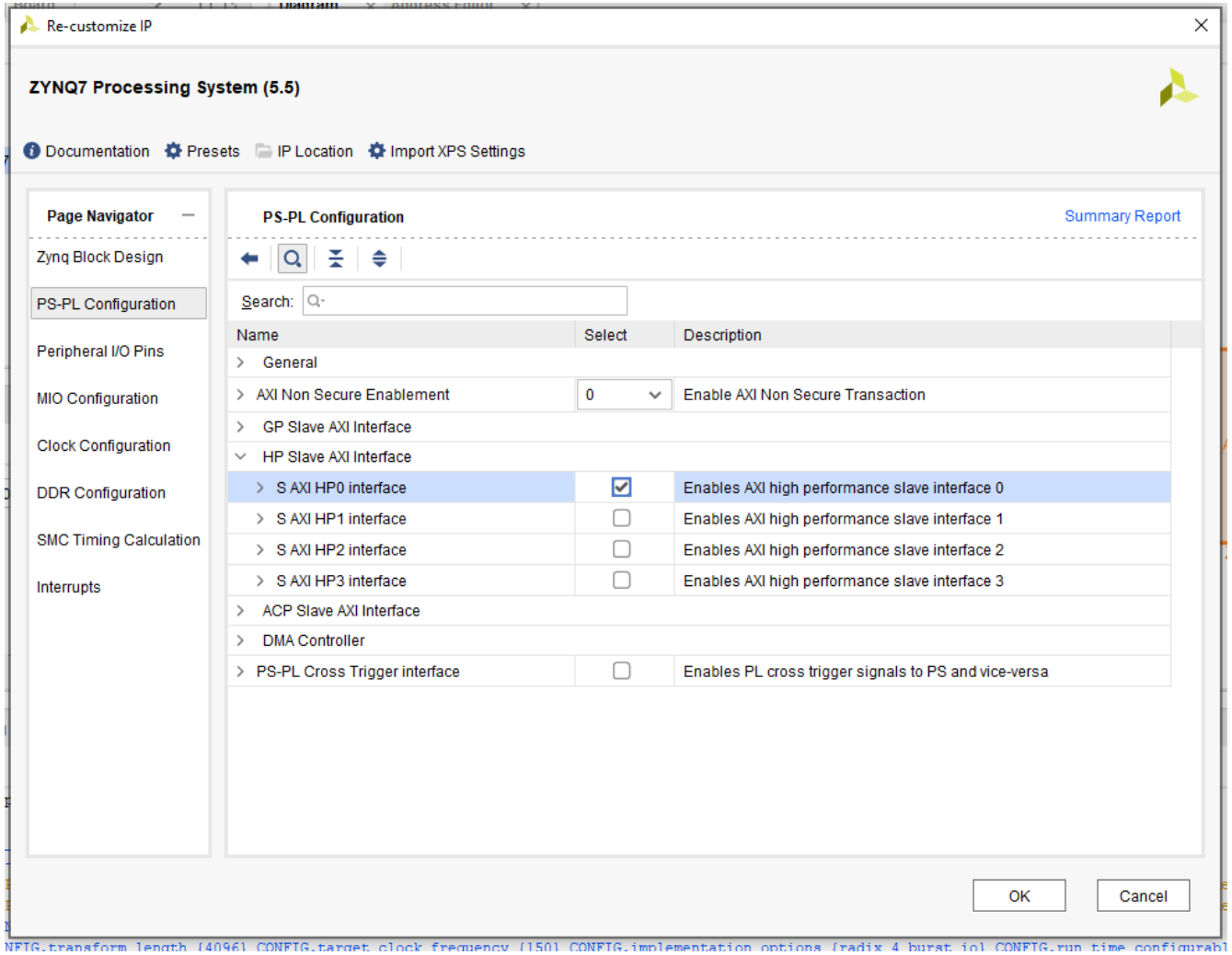


Lab : FFT Acceleration

On the PS/PL interface select the HP

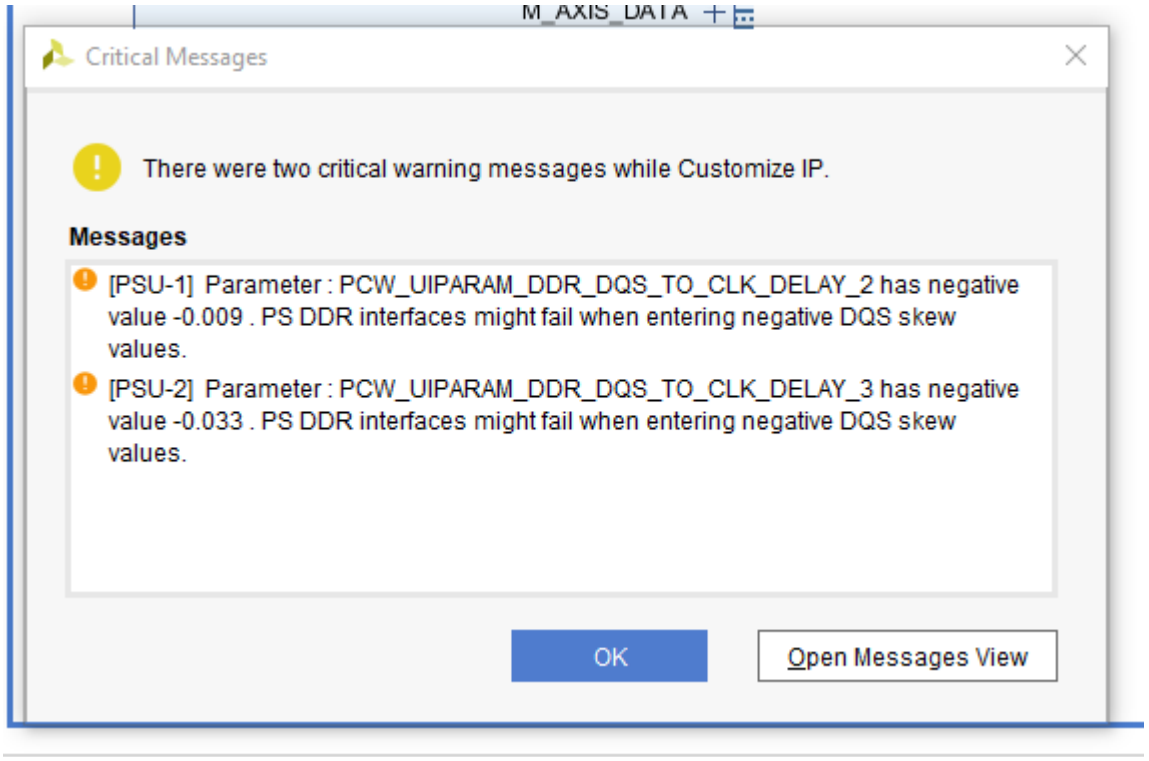
Slave AXI Interface

Enable S AXI HP0 Interface



Lab : FFT Acceleration

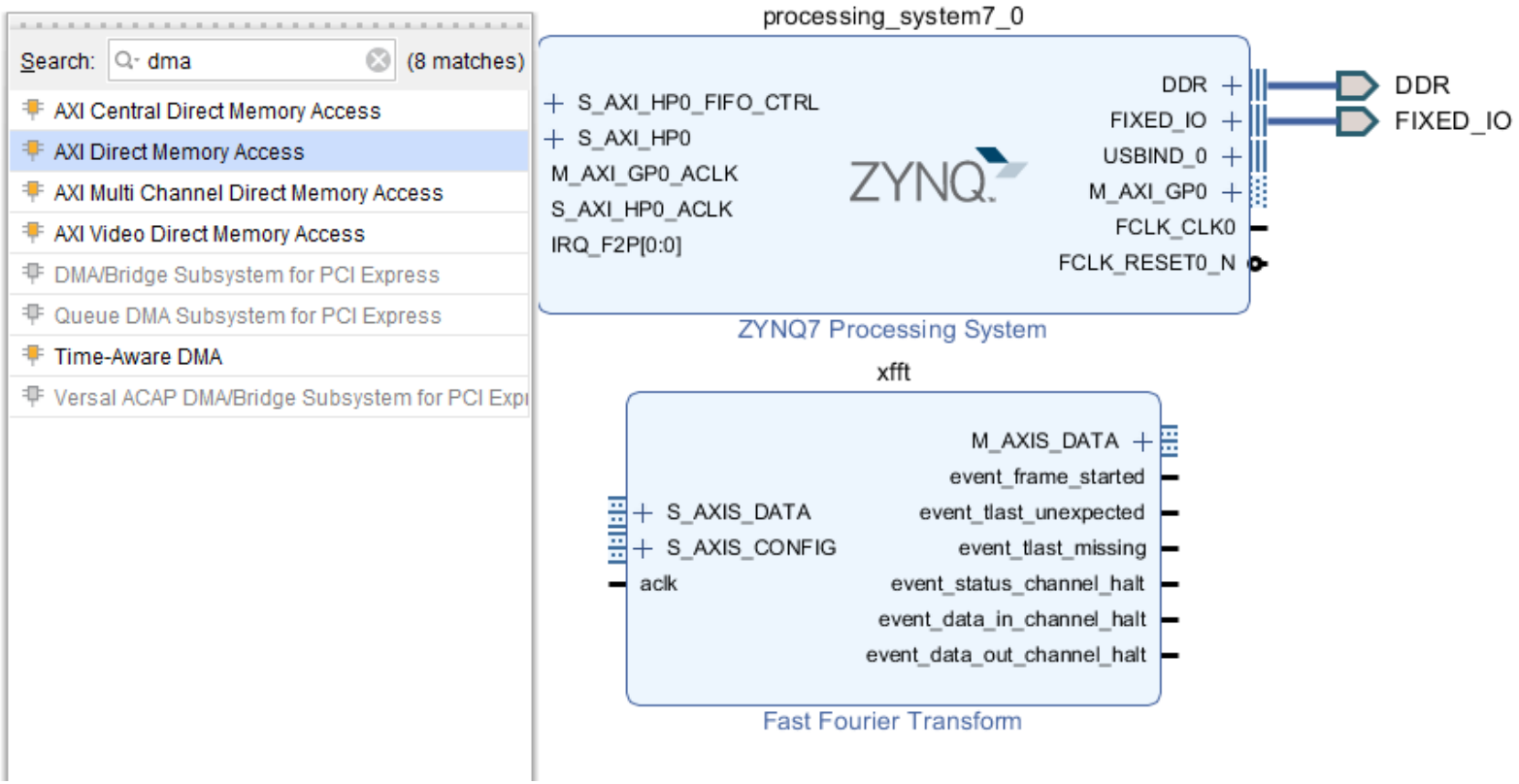
Click OK if the warning appears



Lab : FFT Acceleration

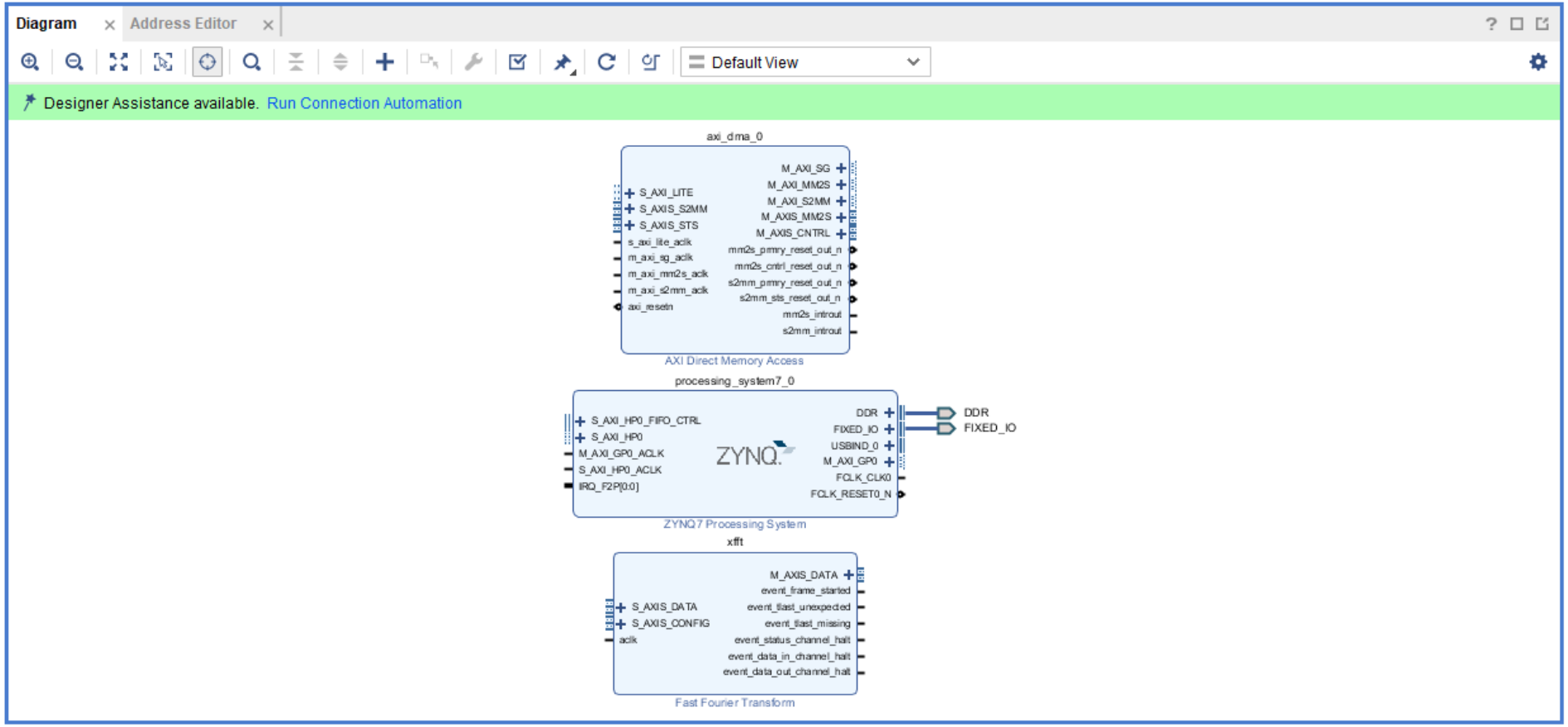
Click + and select AXI

Direct Memory Access



Lab : FFT Acceleration

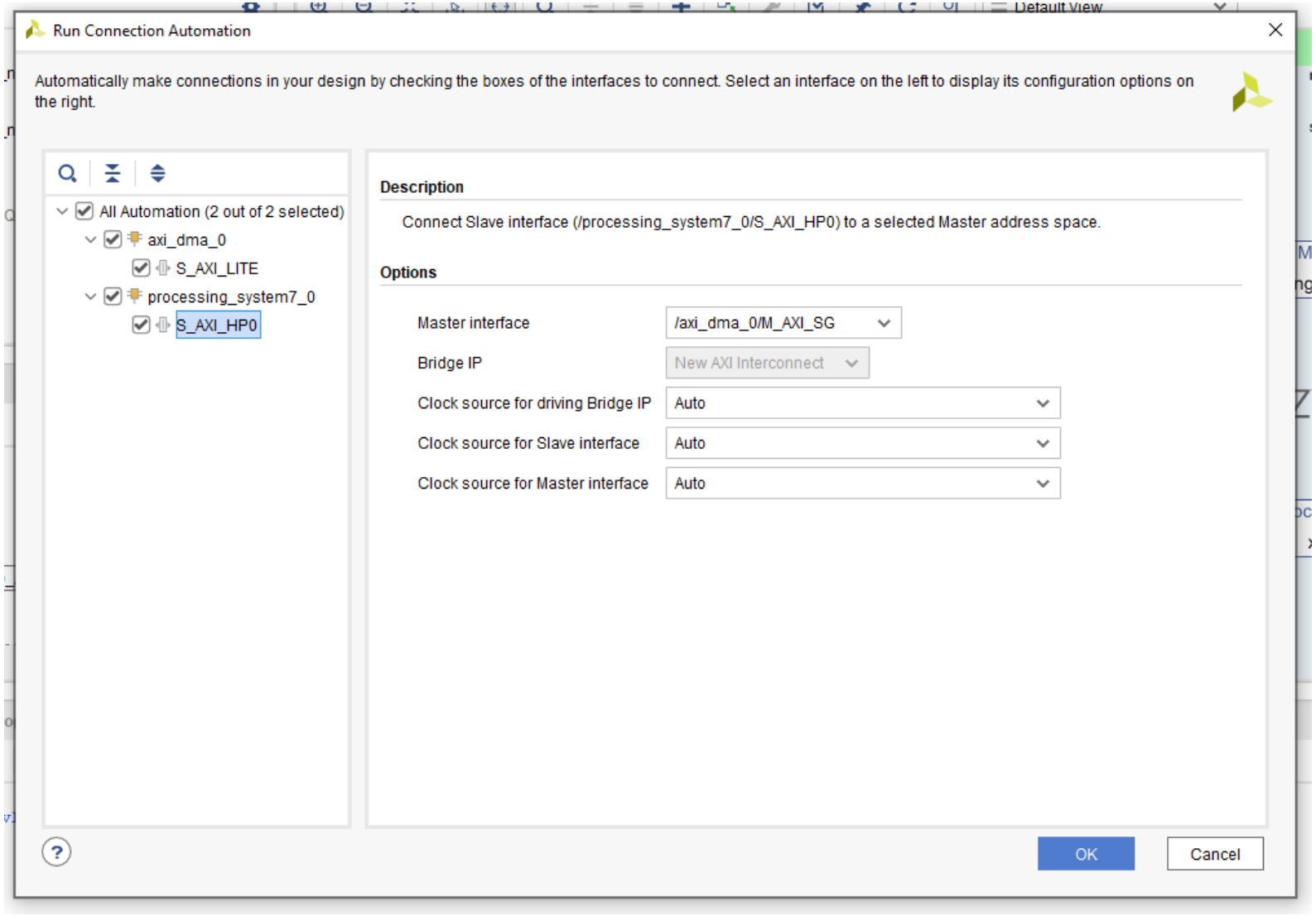
Run the Connection Automation



Lab : FFT Acceleration

Leave the defaults as

standard and click OK

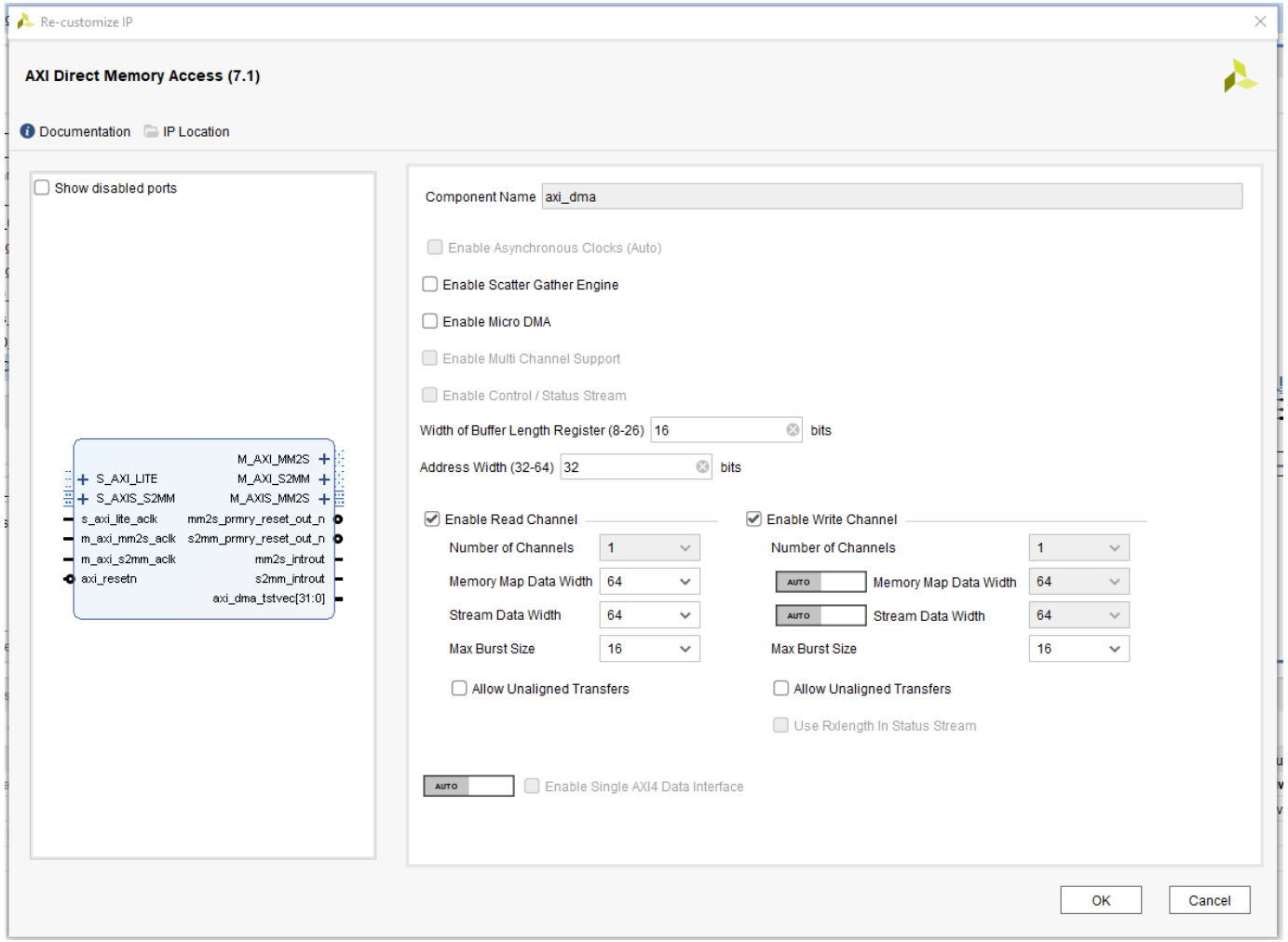


Lab : FFT Acceleration

Select the DMA, double click on it

and configure it

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



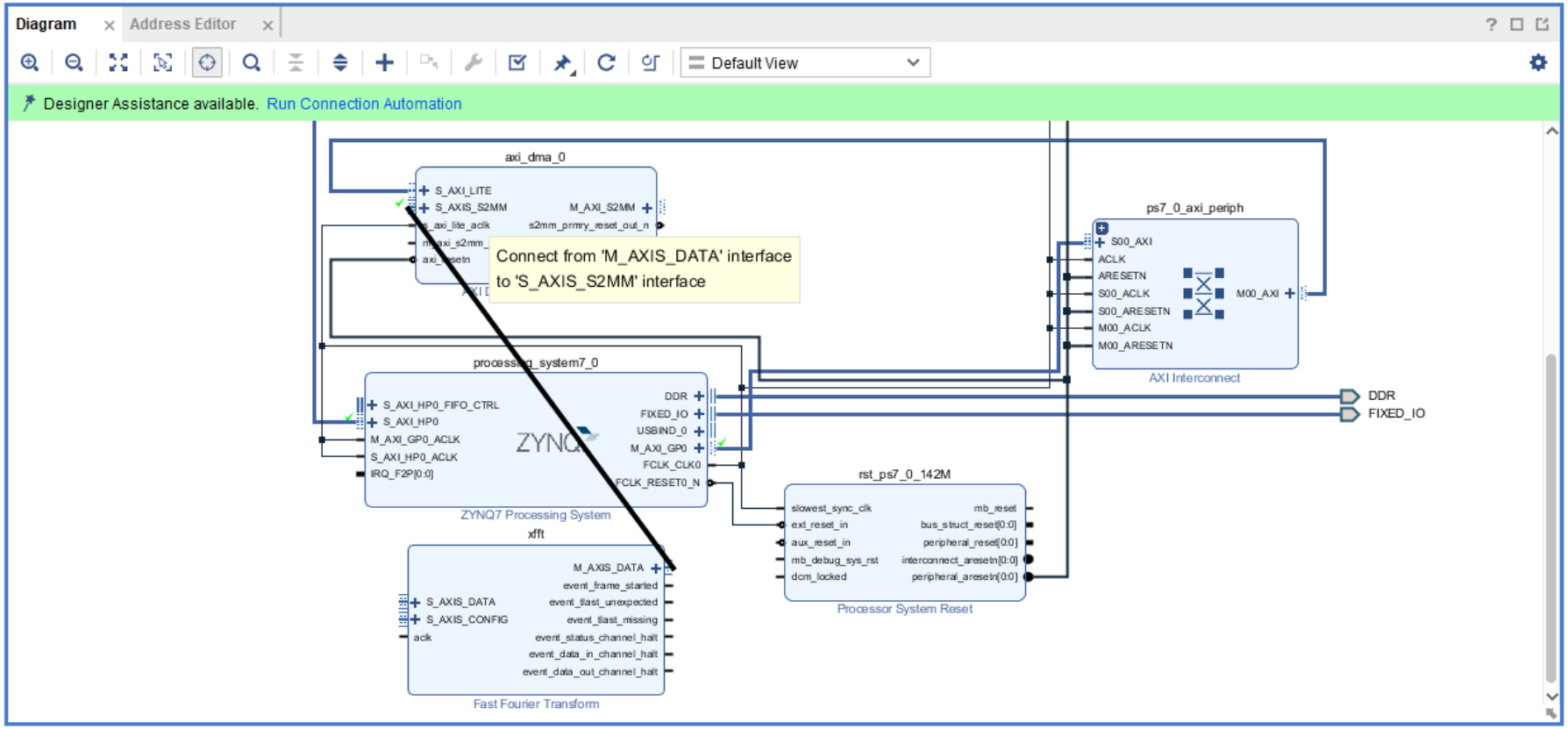
Lab : FFT Acceleration

Connect the xFFT M

AXIS data to the

DMA, S AXIS S2MM

port

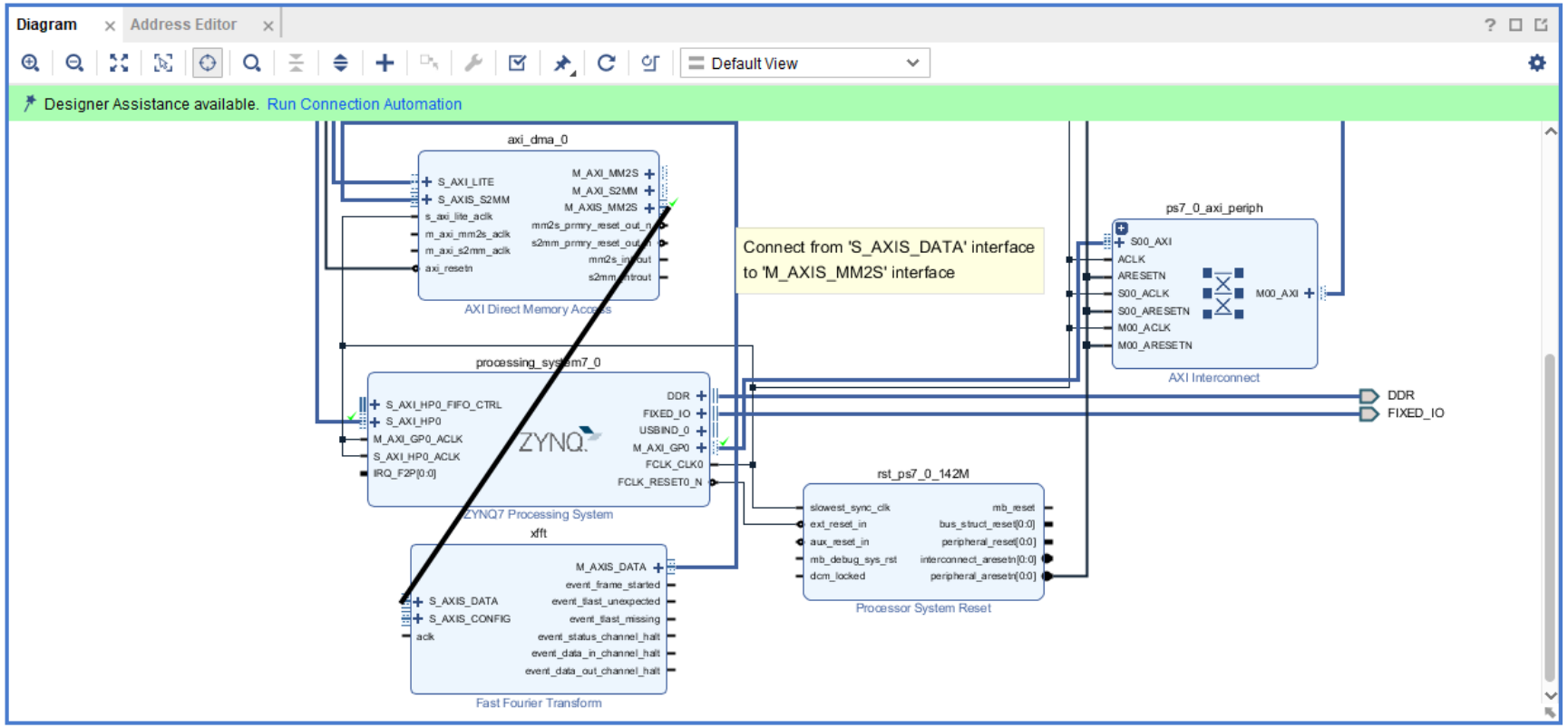


Lab : FFT Acceleration

Connect the DMA M

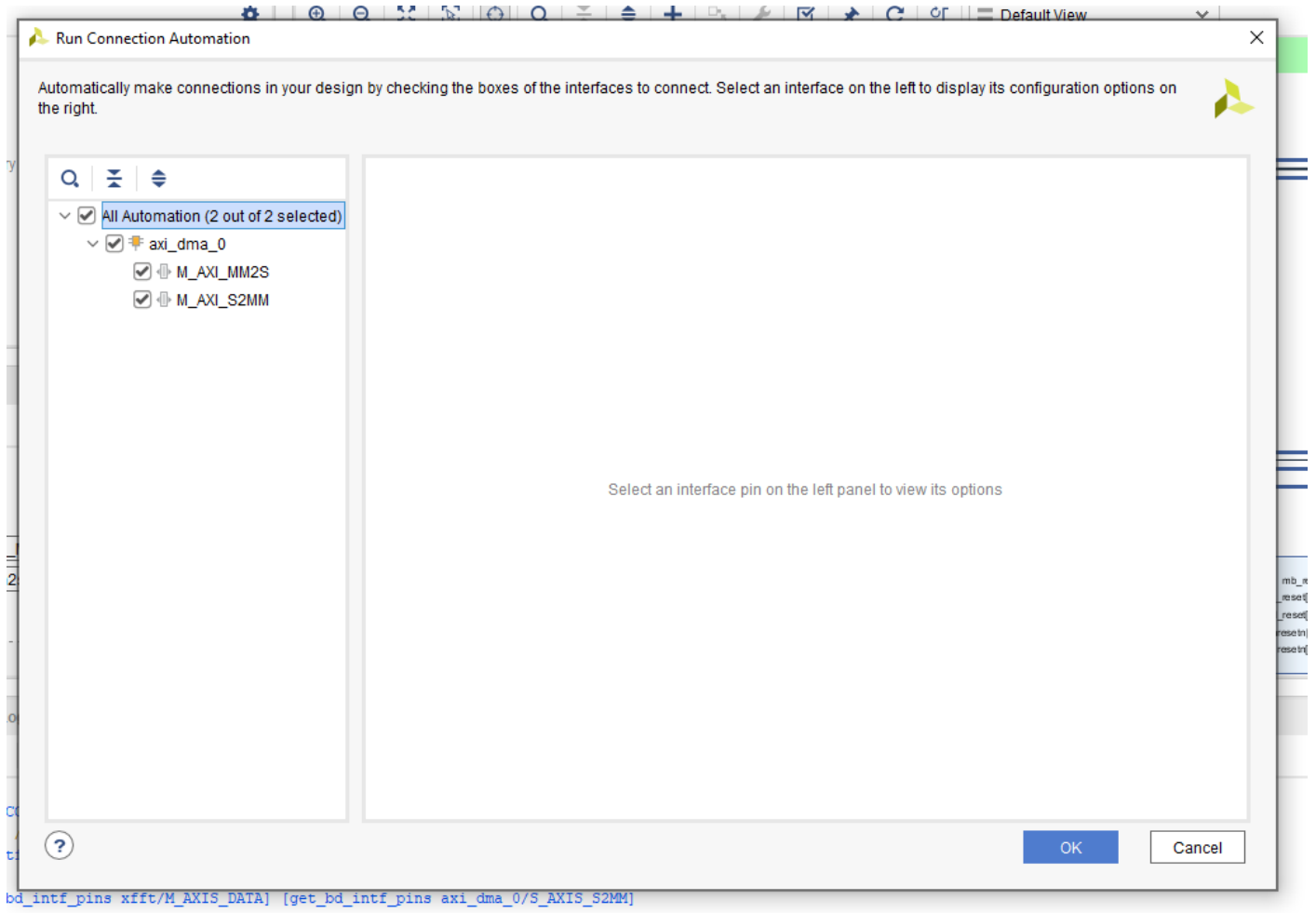
AXIS MM2S to the xFFT

S AXIS Data



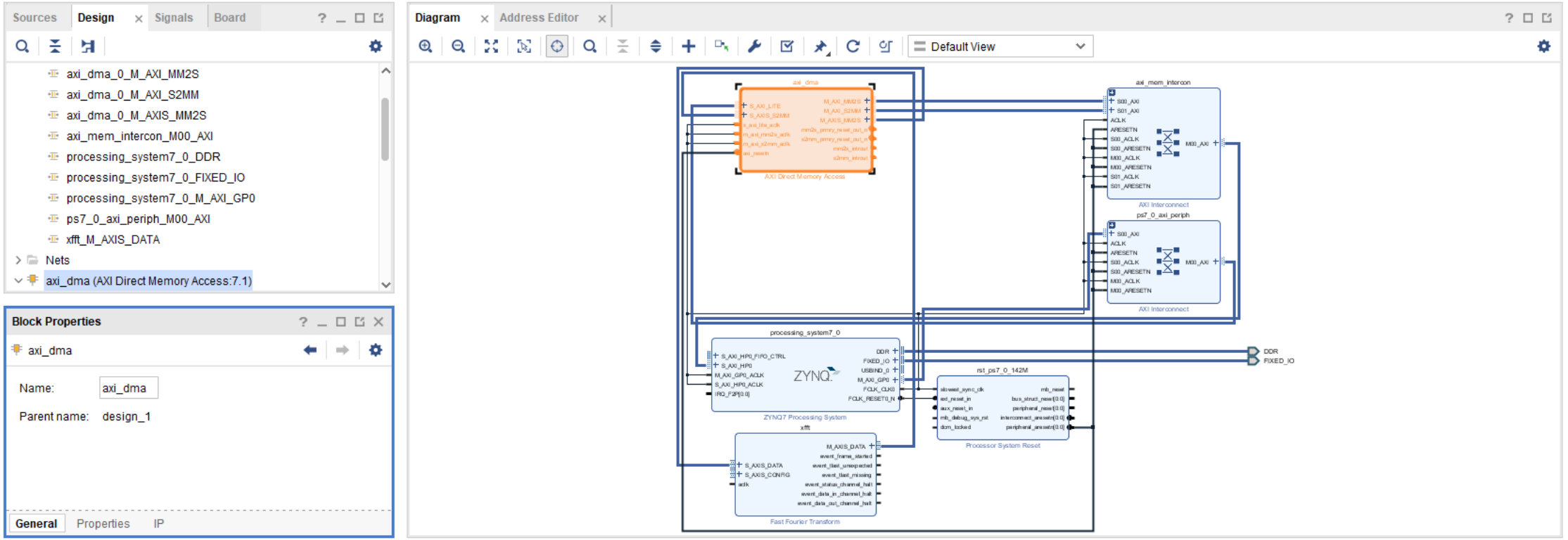
Lab : FFT Acceleration

Run the connection automation



Lab : FFT Acceleration

The diagram should look like below

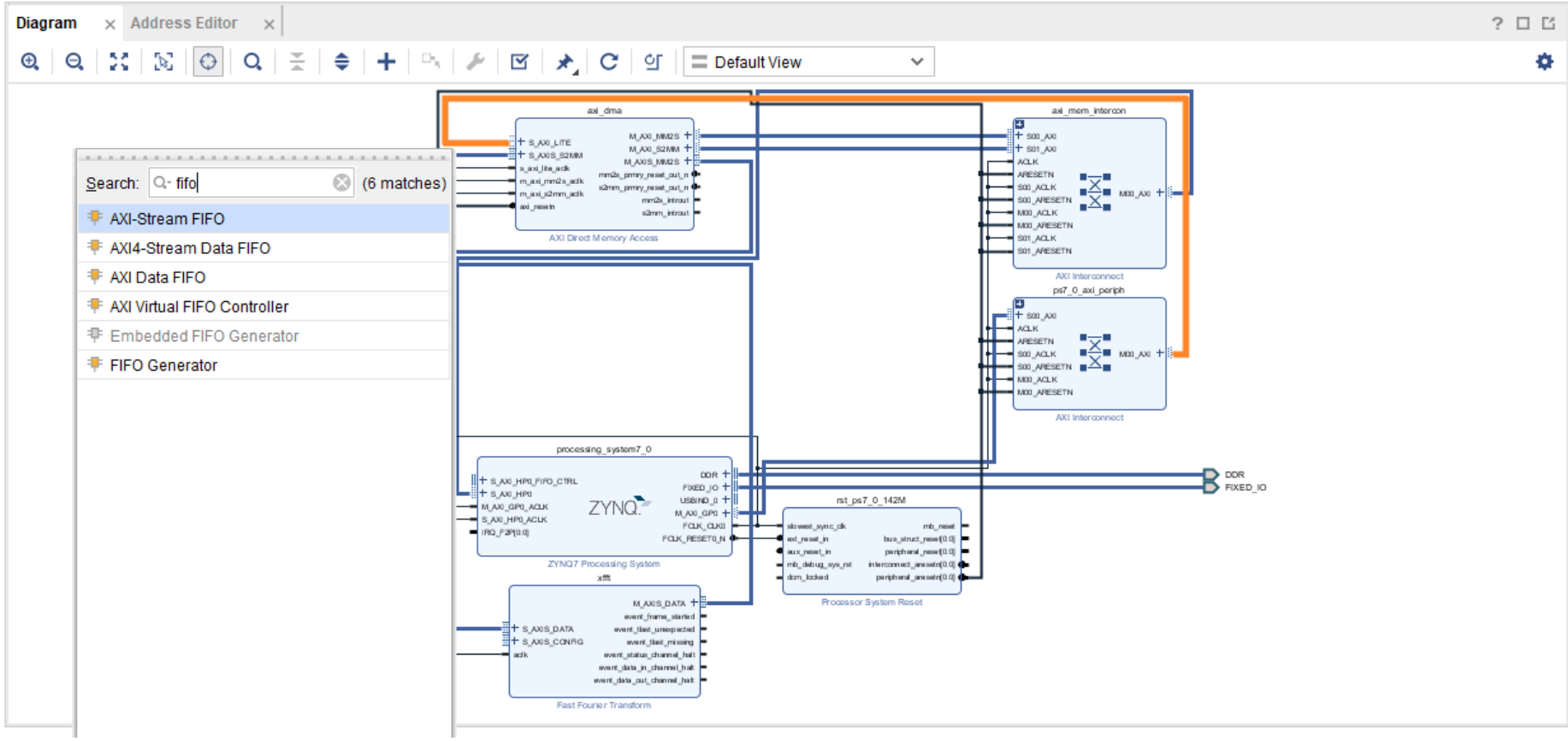


Lab : FFT Acceleration

Click on + and add

in an AXI-Stream

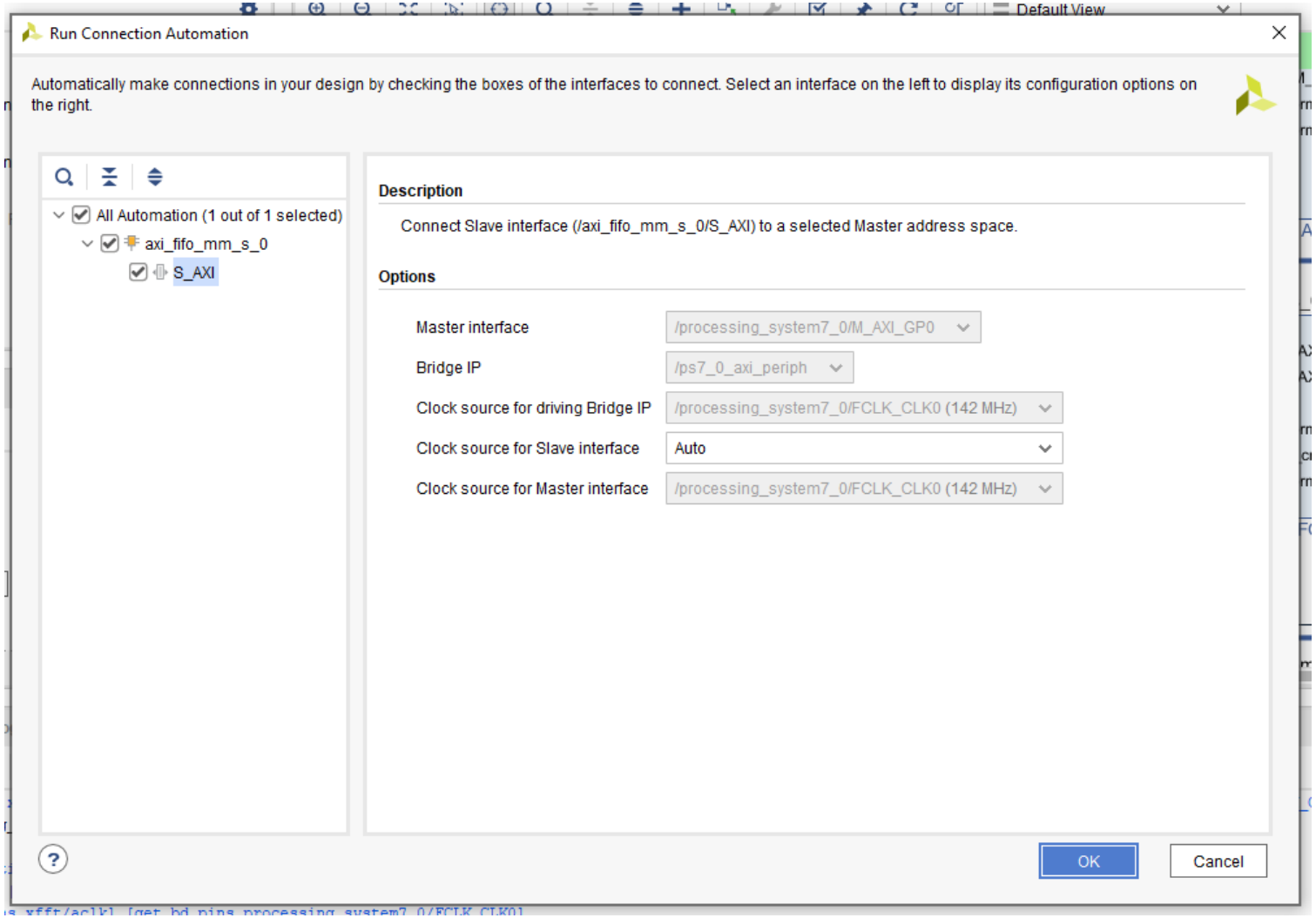
FIFO



Lab : FFT Acceleration

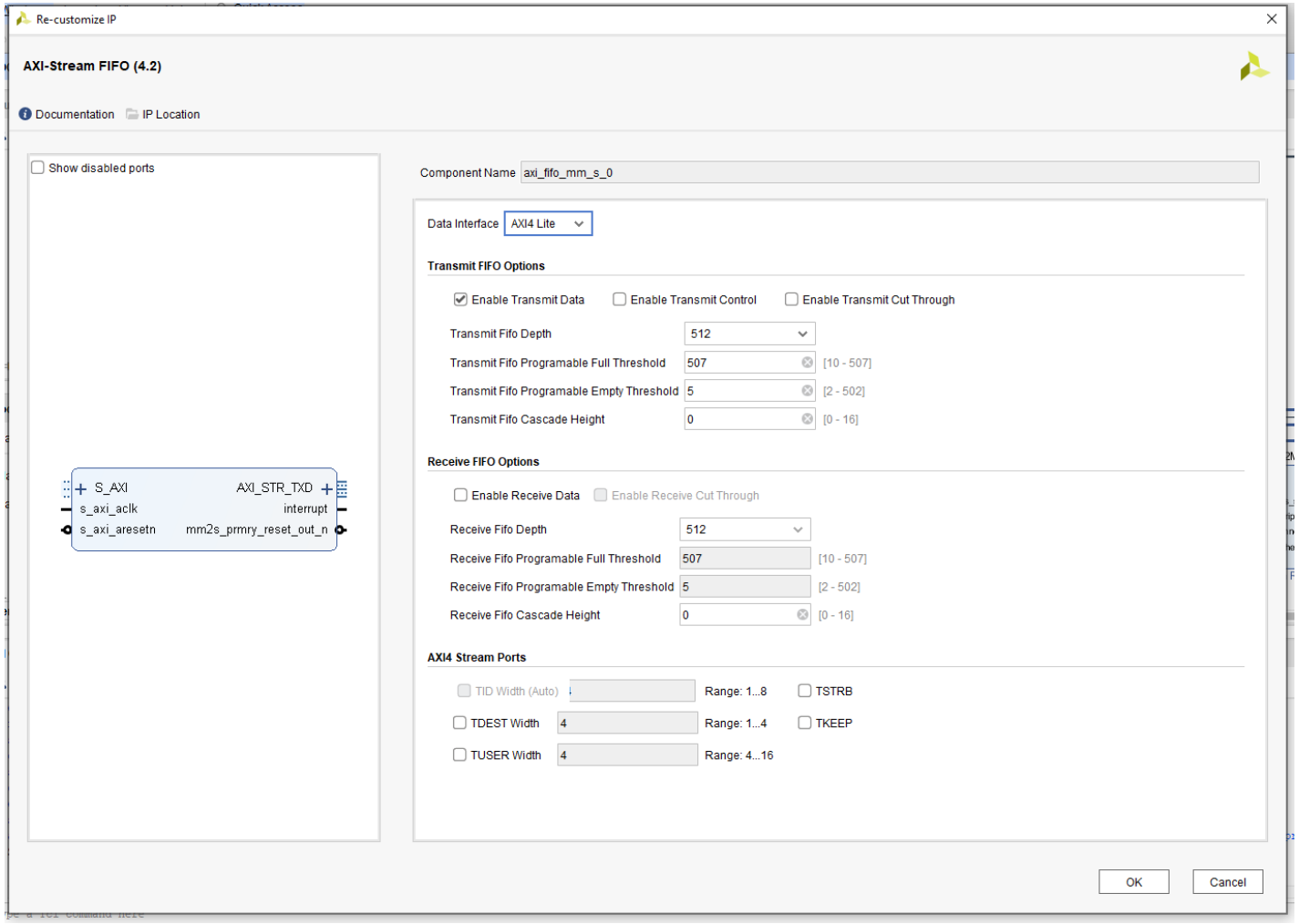
Run the connection

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.

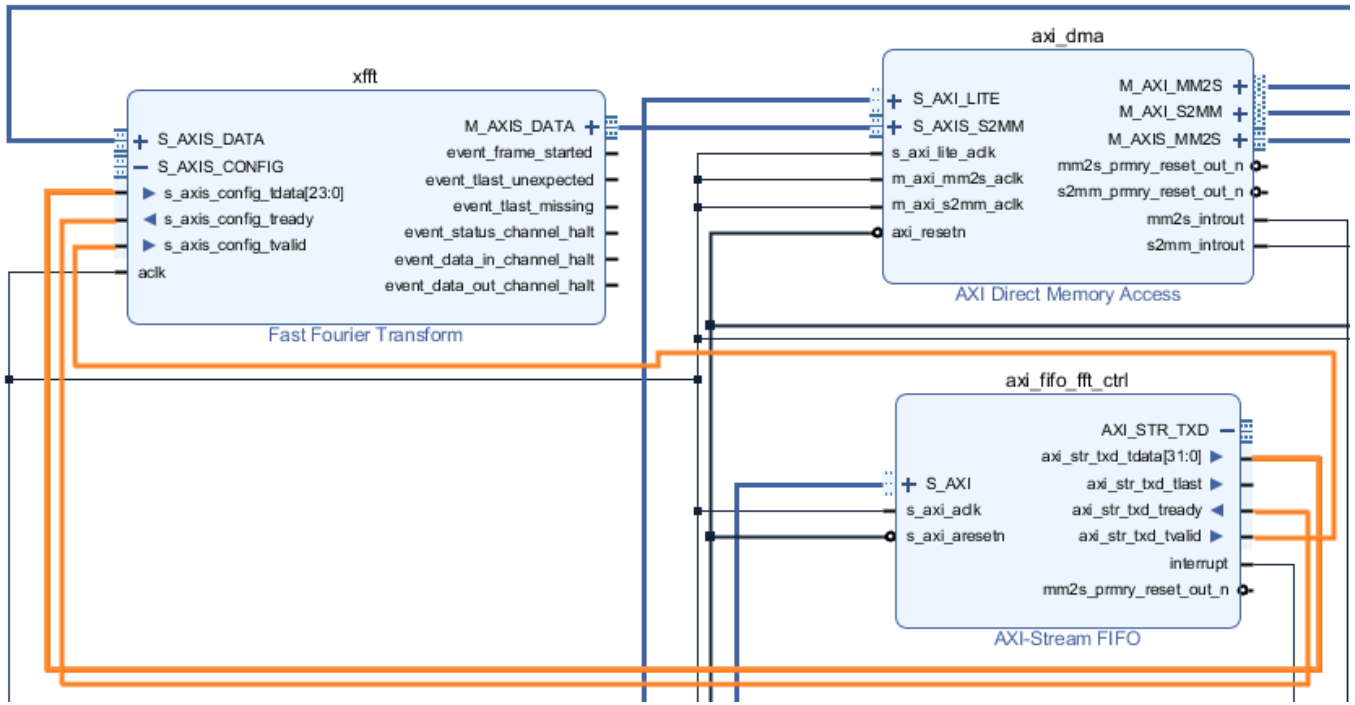


Lab : FFT Acceleration

Connect the AXI STR TXD tdata,

tlast and tvalid signals to the

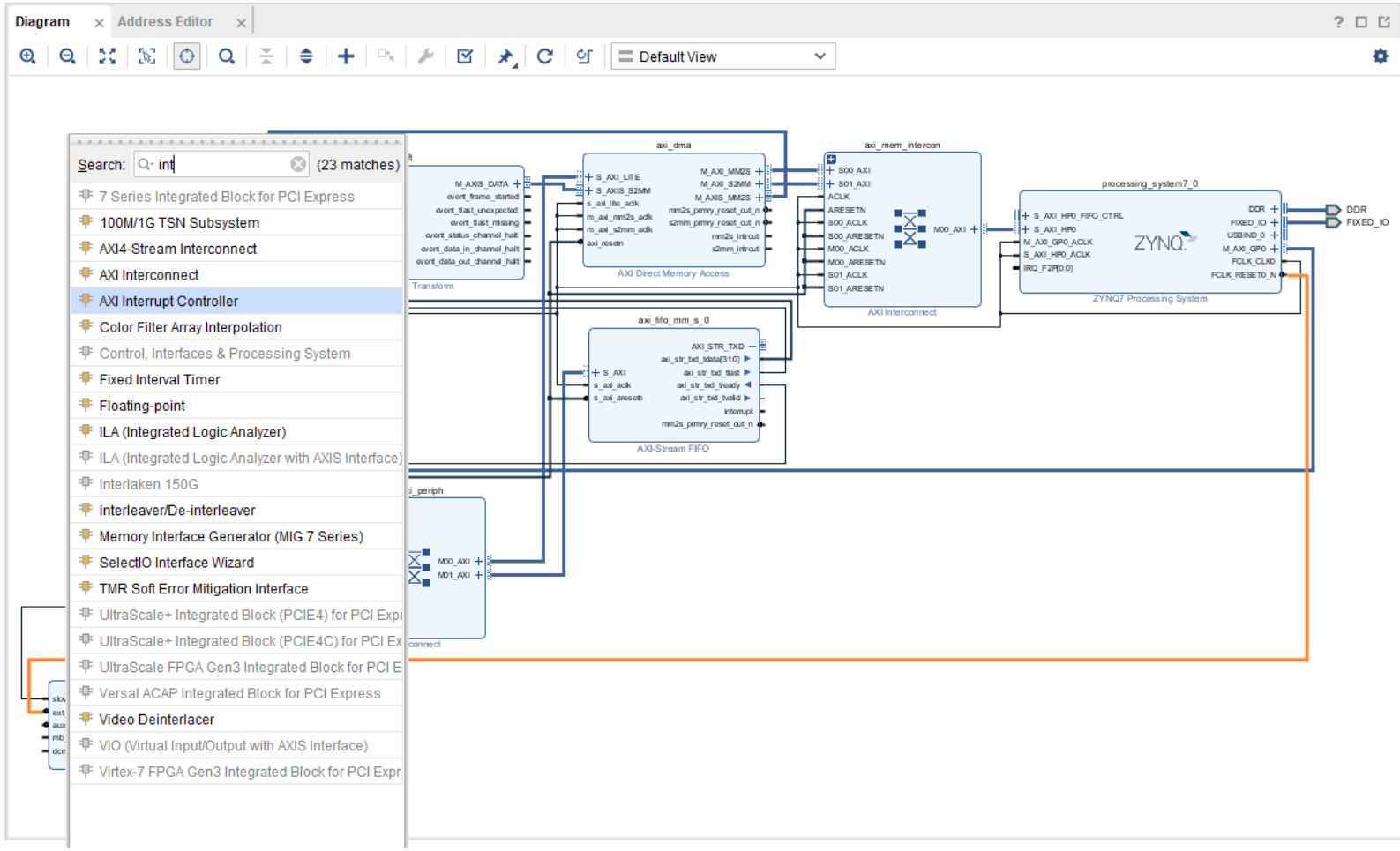
xFFT S AXIS config



Lab : FFT Acceleration

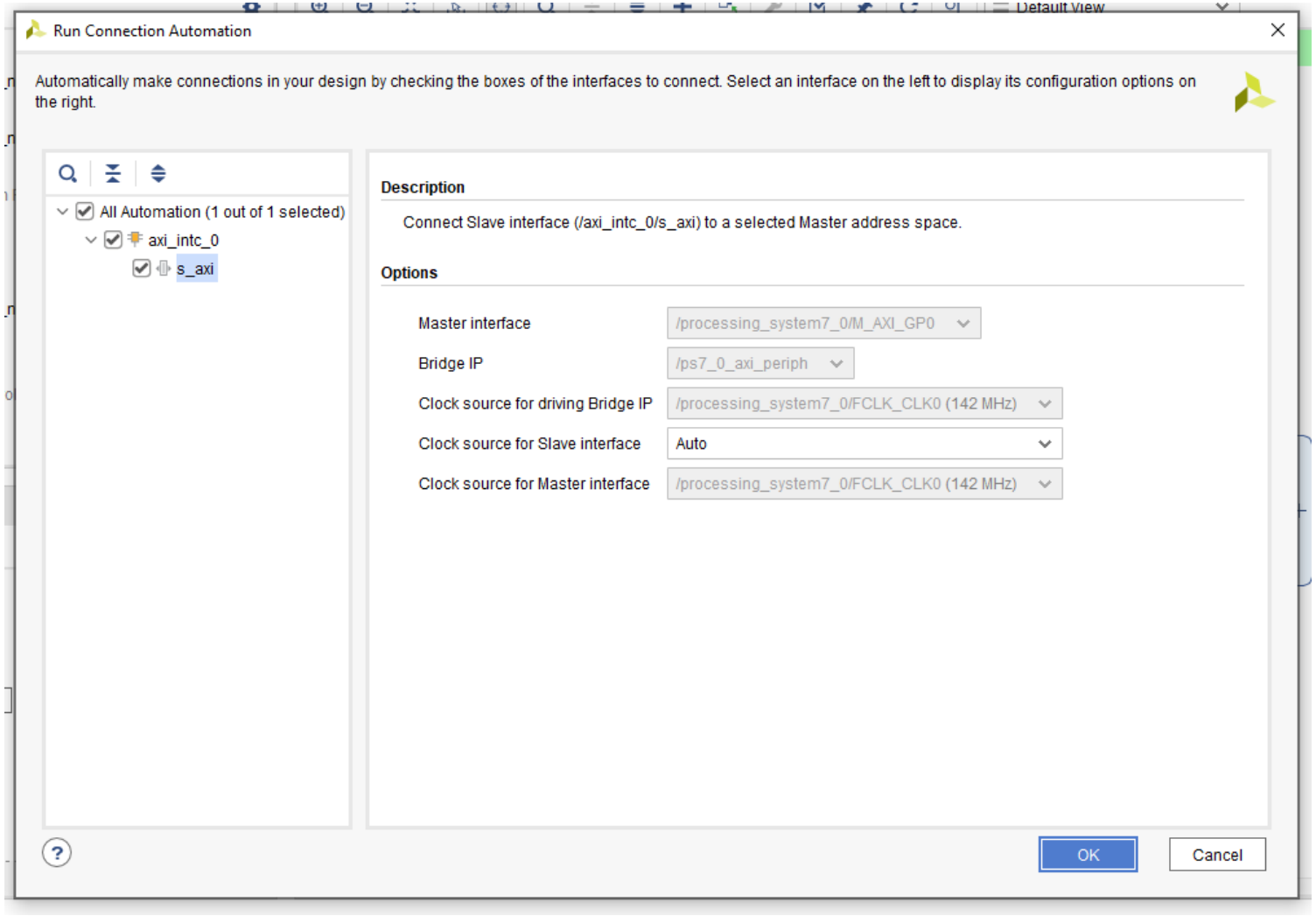
Click on + and add in a AXI

Interrupt Controller



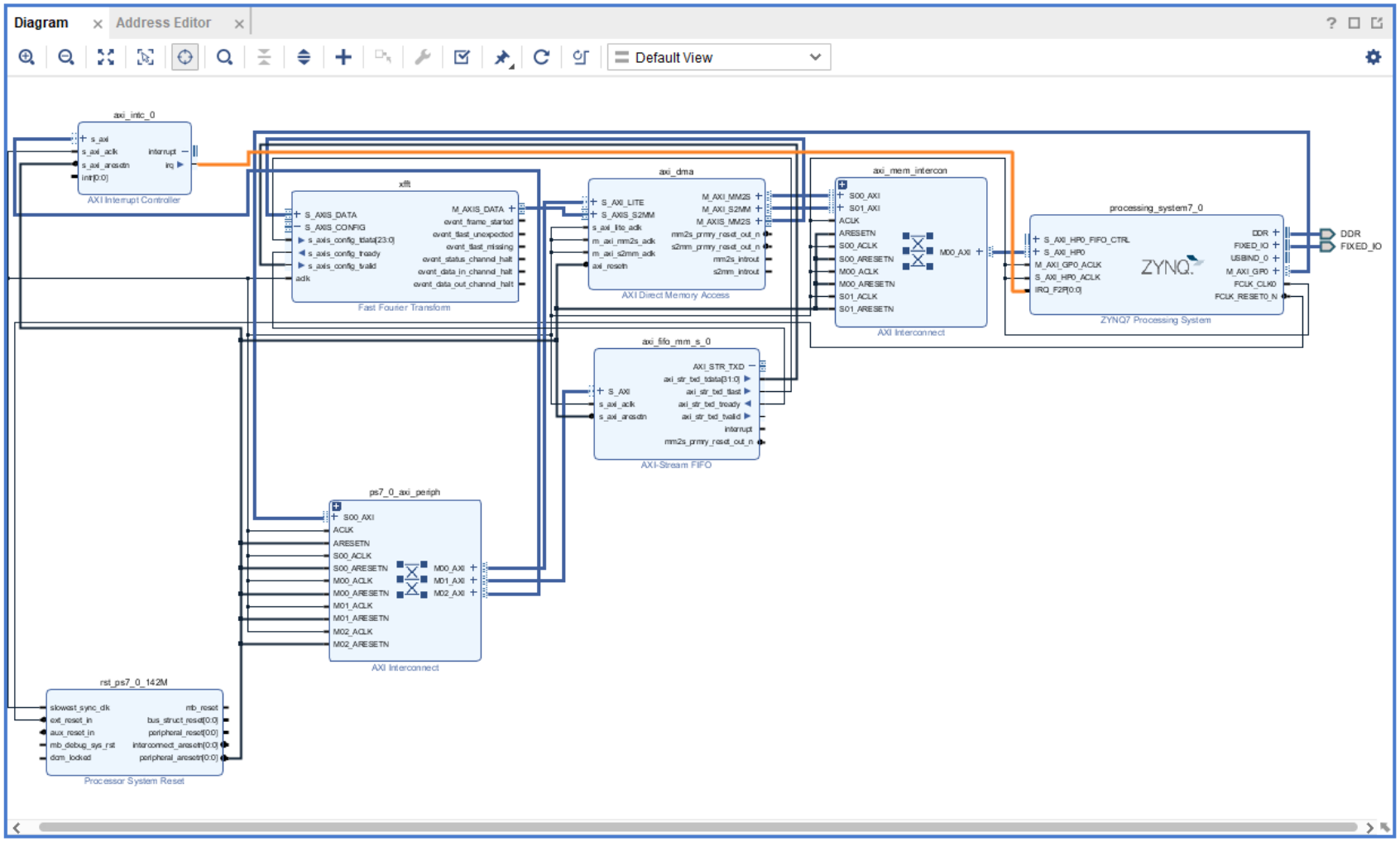
Lab : FFT Acceleration

Run the connection automation



Lab : FFT Acceleration

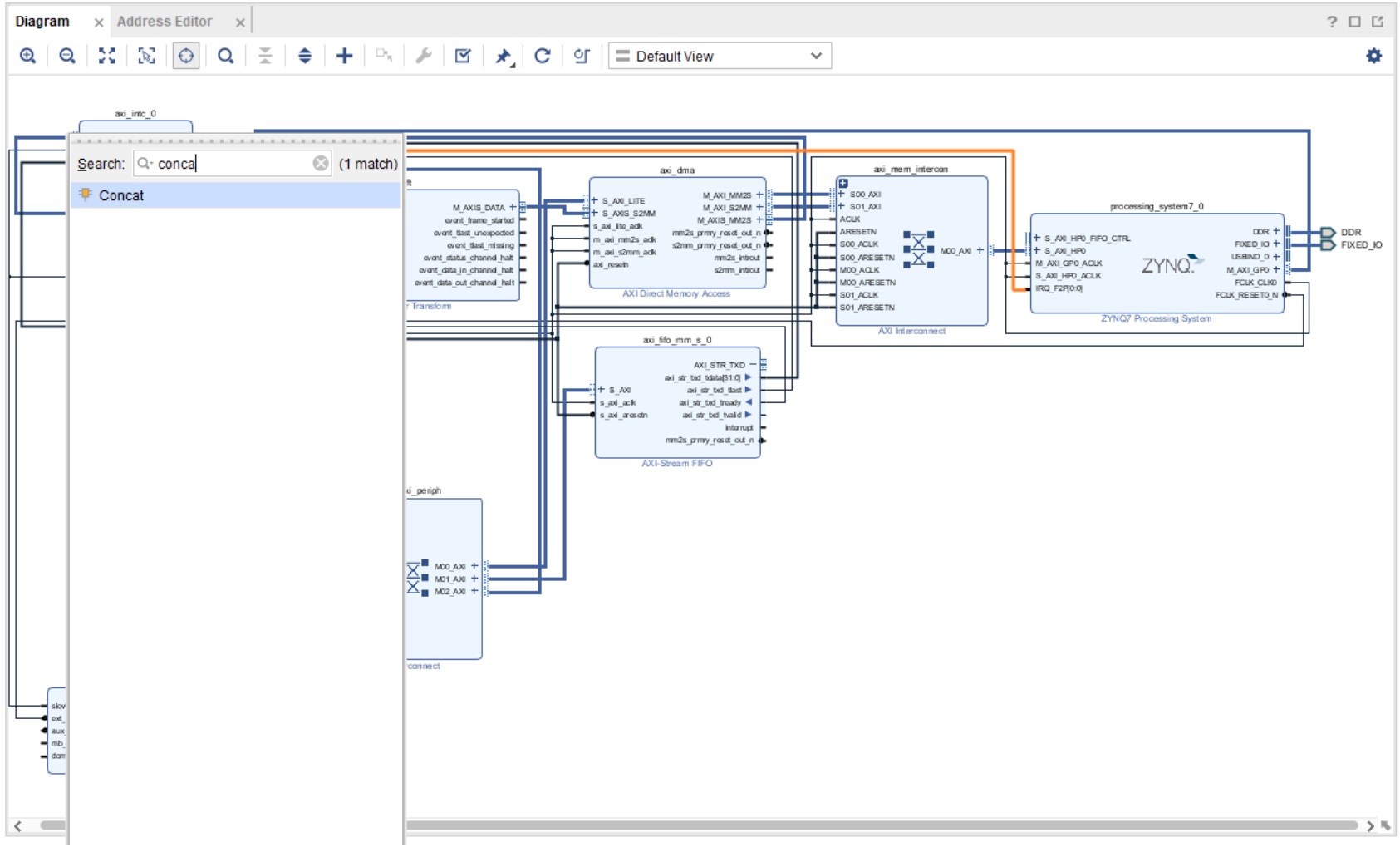
Connect the IRQ output from the AXI Interrupt Controller to the IRQF2P port on the Zynq



Lab : FFT Acceleration

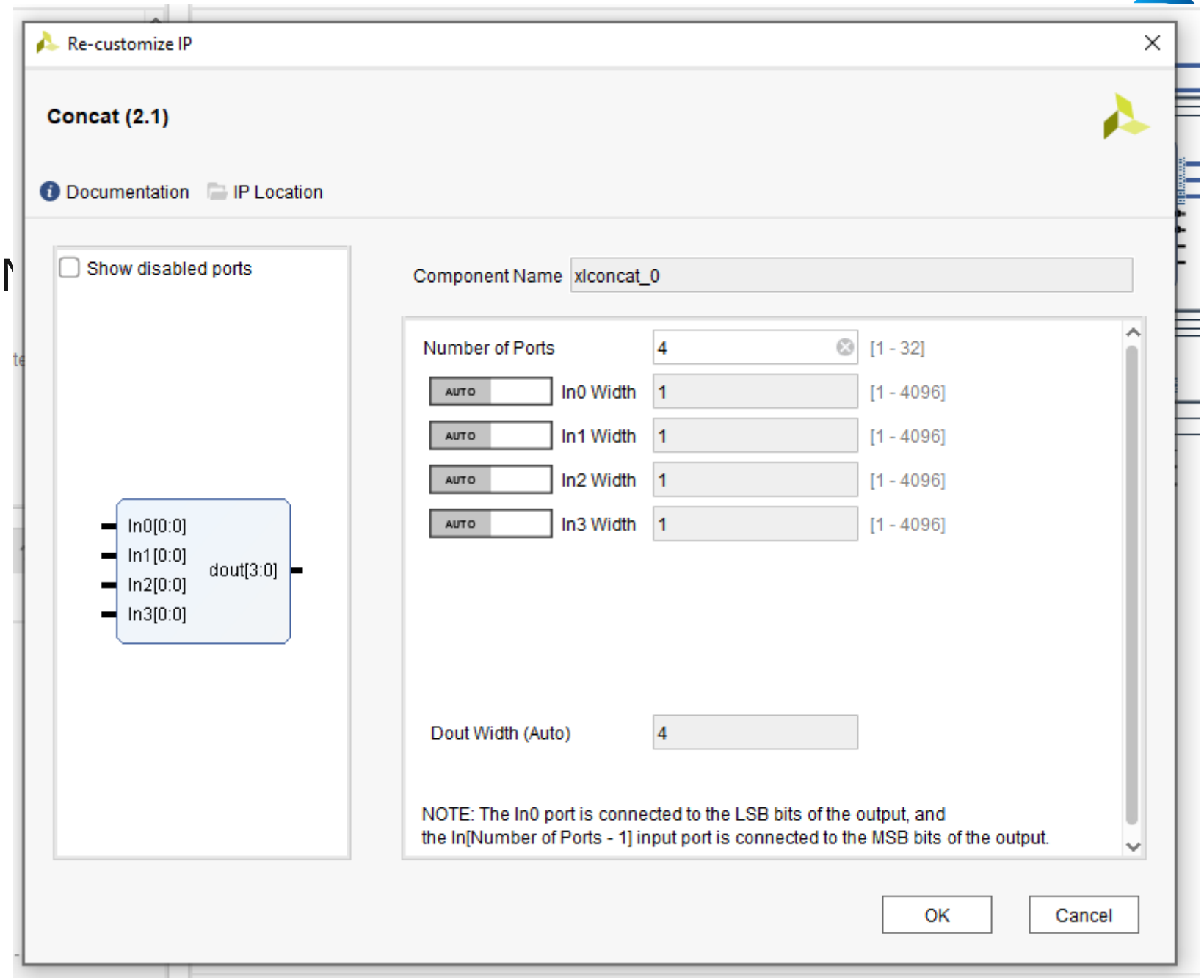
Click on + and add in a

concat block



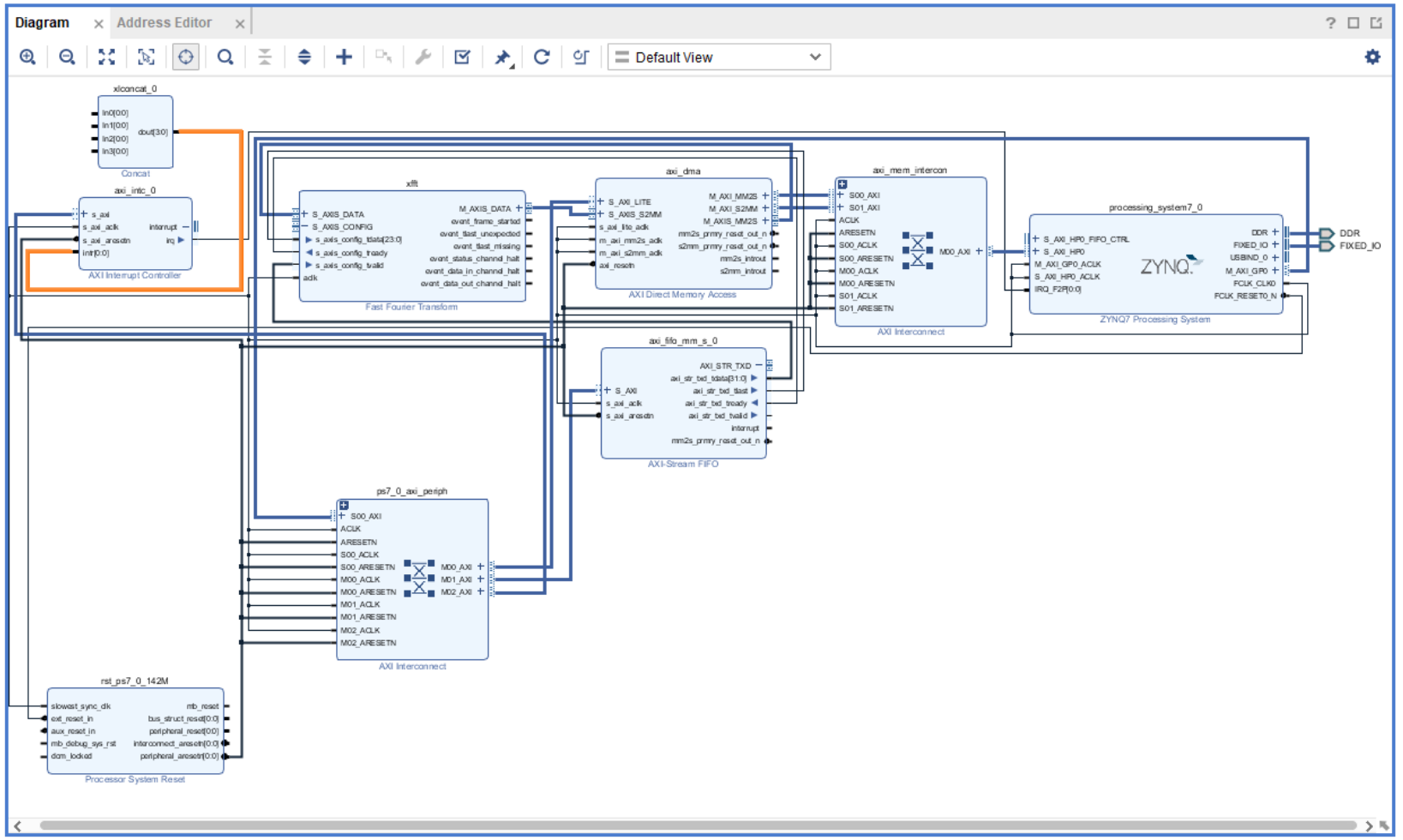
Lab :

Go to PYN


Re-customize IP
Concat (2.1)
Documentation
IP Location
Show disabled ports
In0[0:0]
In1[0:0]
In2[0:0]
In3[0:0]
dout[3:0]
Component Name
xlconcat_0
Number of Ports
4
[1 - 32]
AUTO
In0 Width
1
[1 - 4096]
In1 Width
In2 Width
In3 Width
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

Lab : FFT Acceleration

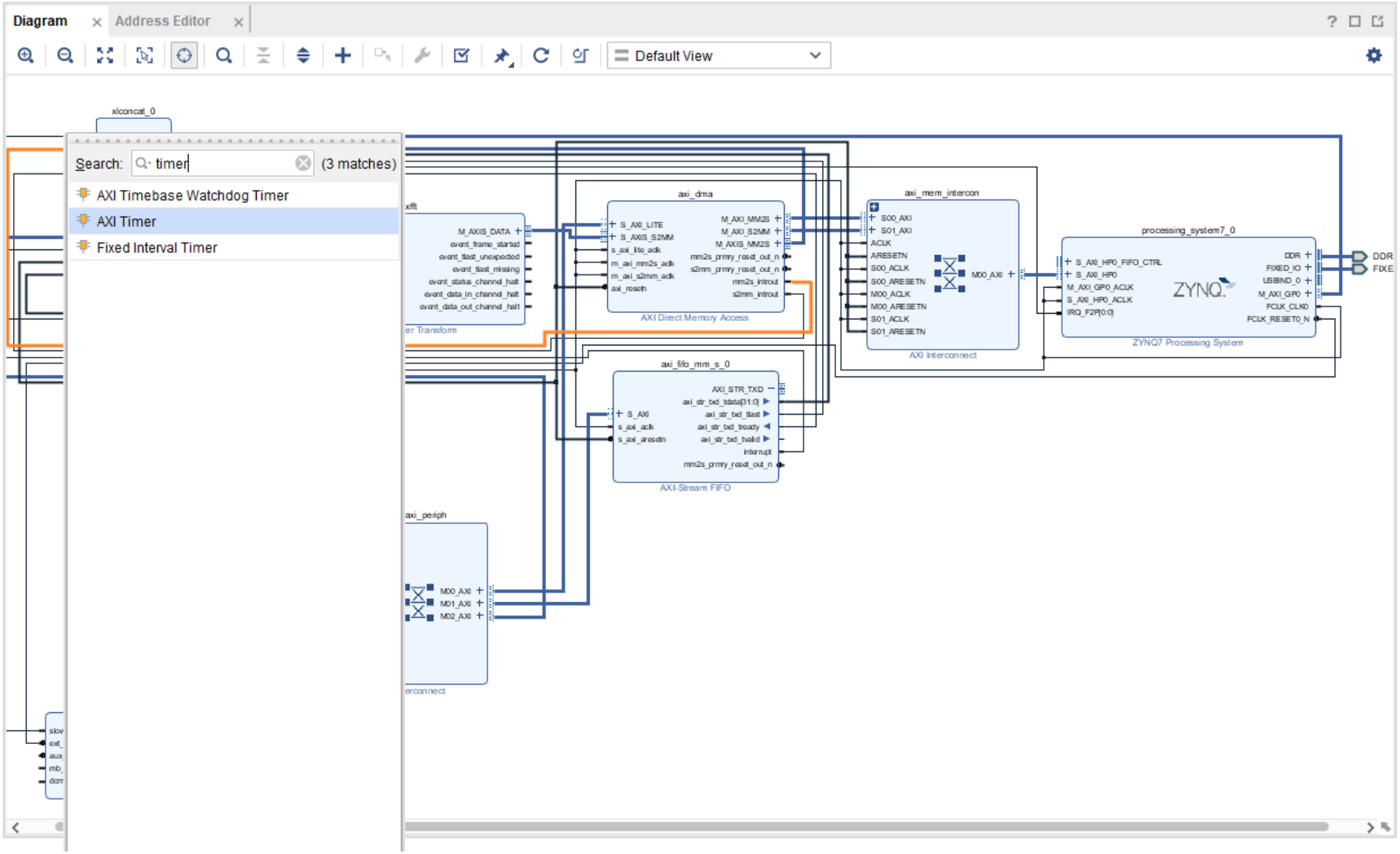
Connect the output of the concat block to the AXI Interrupt controller INT input



Lab : FFT Acceleration

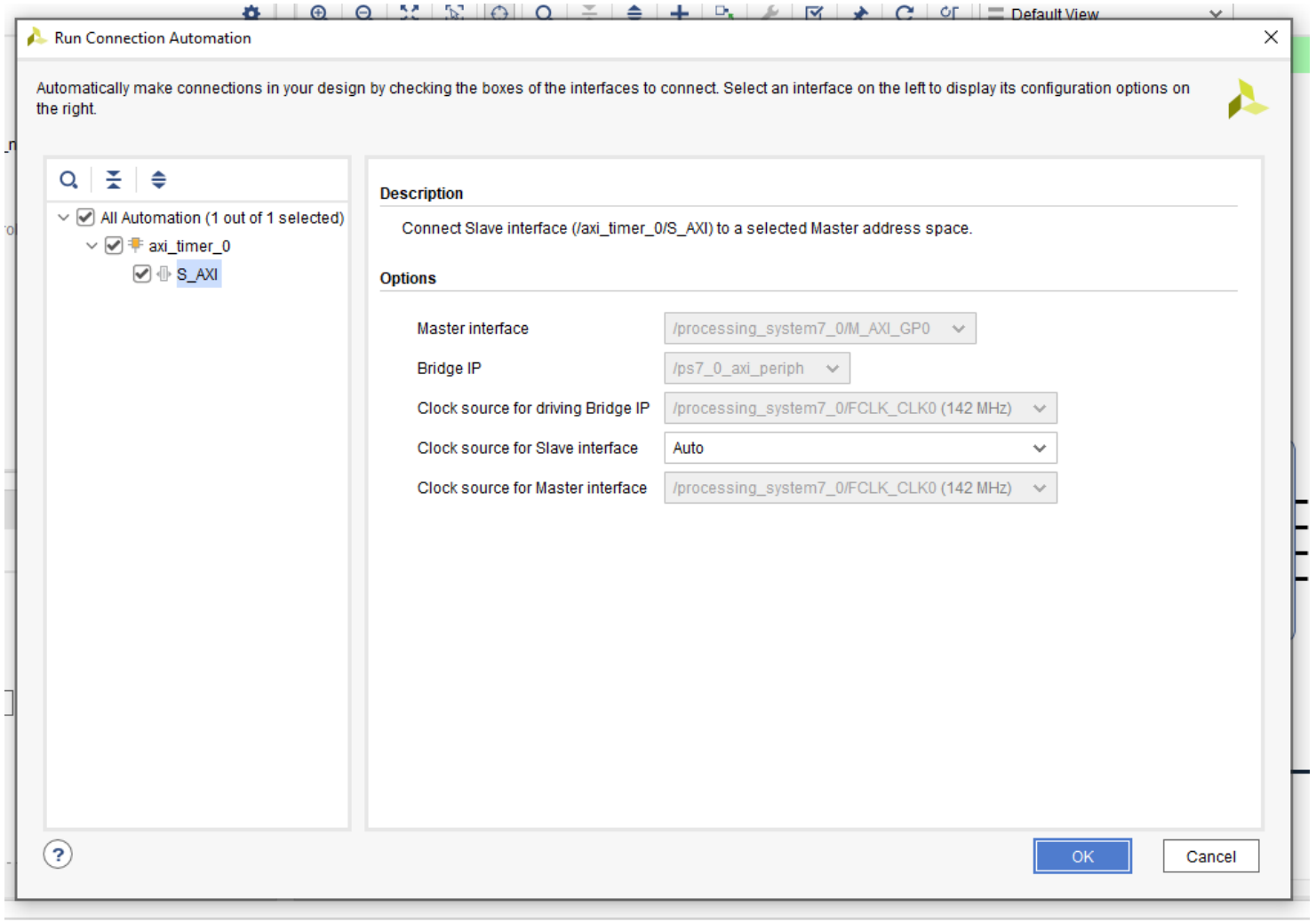
Click on the + symbol

and add in a AXI Timer



Lab : FFT Acceleration

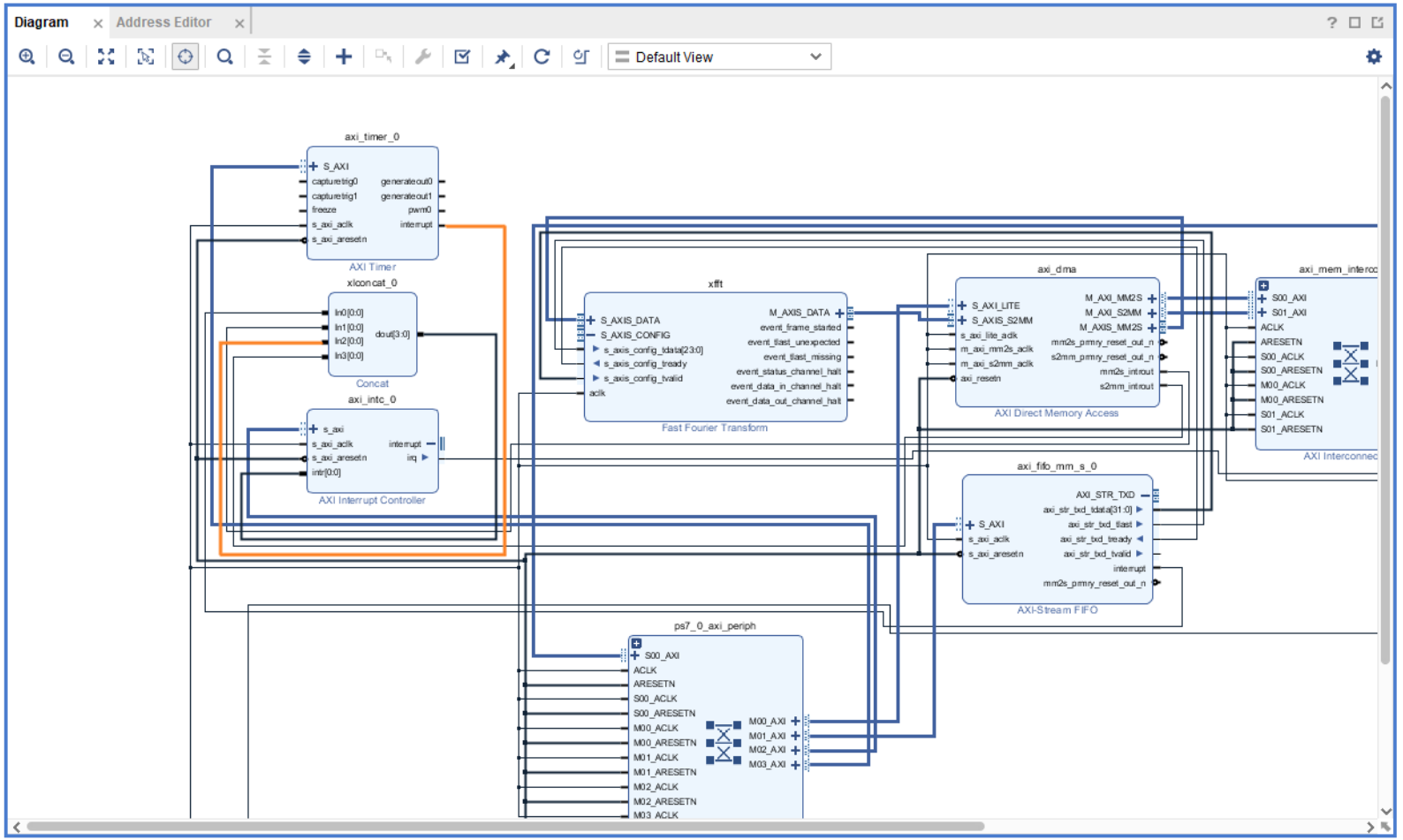
Run the connection automation



Lab : FFT Acceleration

Connect the interrupts to the concat block

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



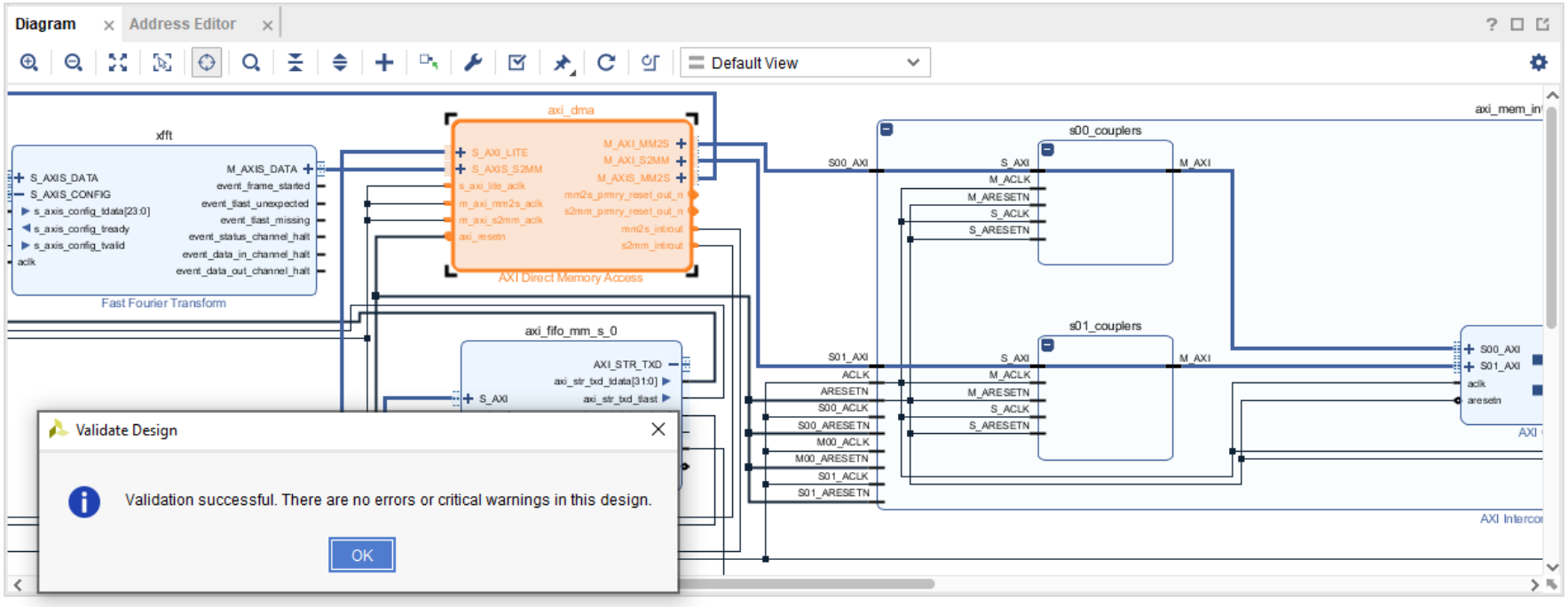
Lab : FFT Acceleration

Validate the design

there should be no

error or critical

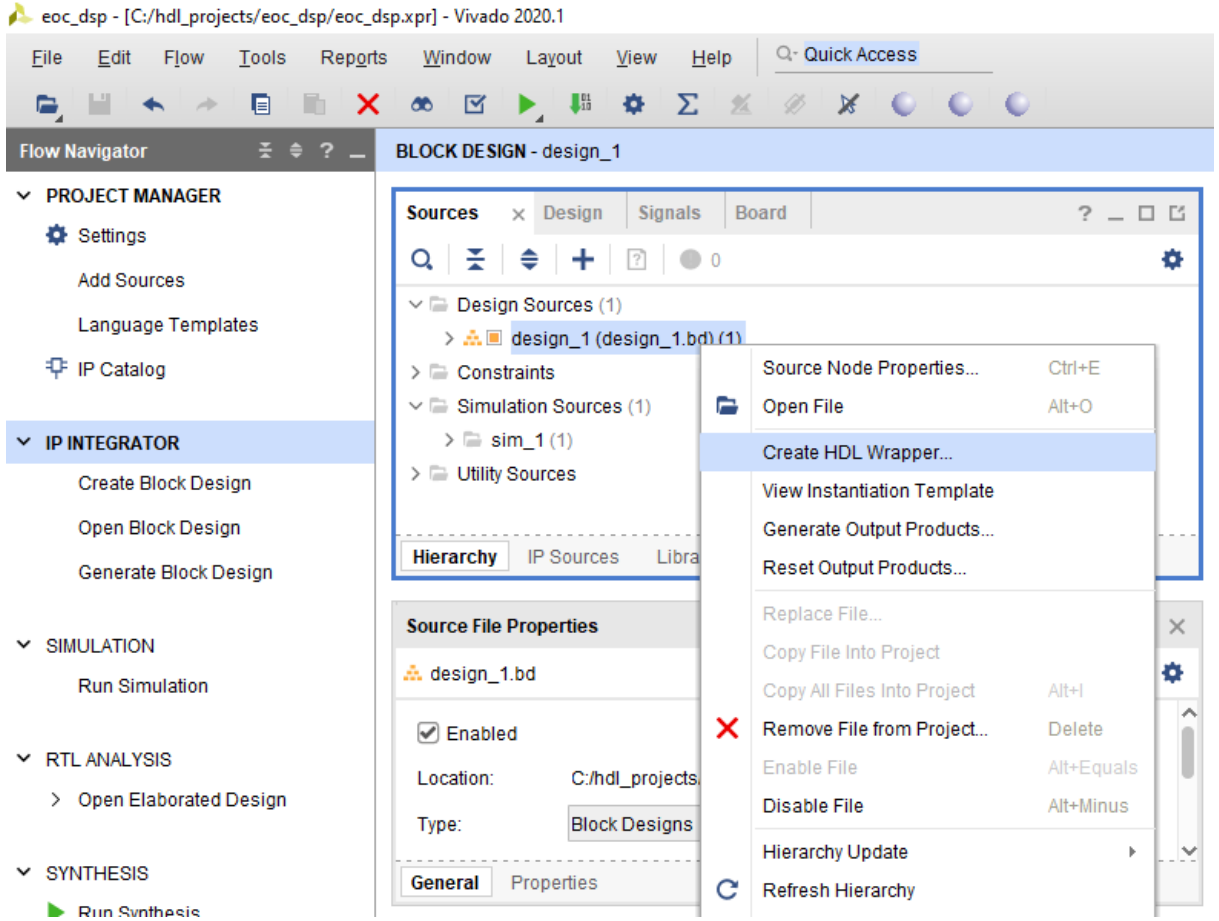
warnings



Lab : FFT Acceleration

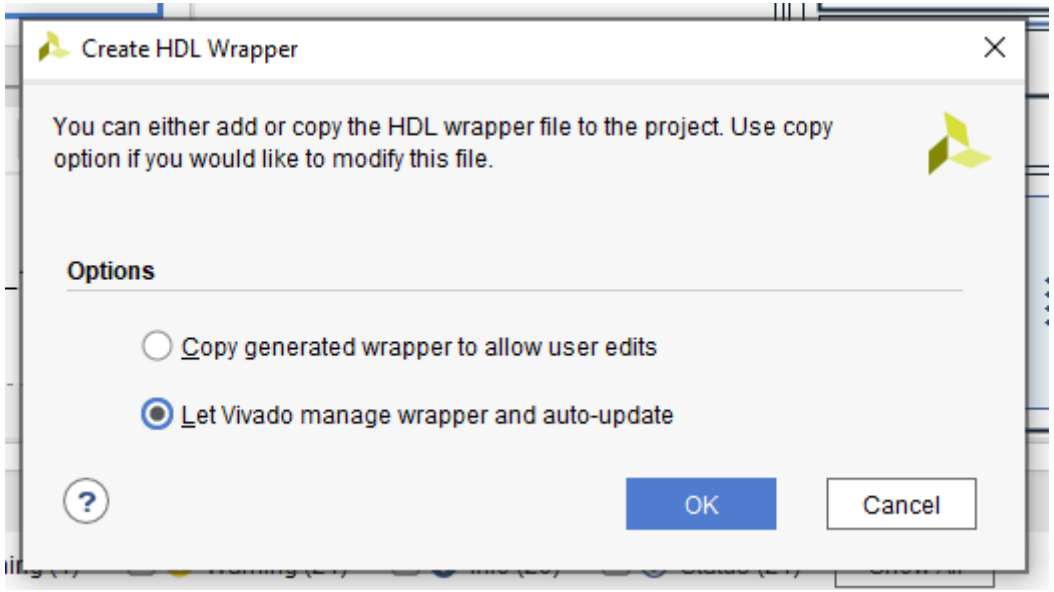
Right click on the design and

select Create HDL Wrapper



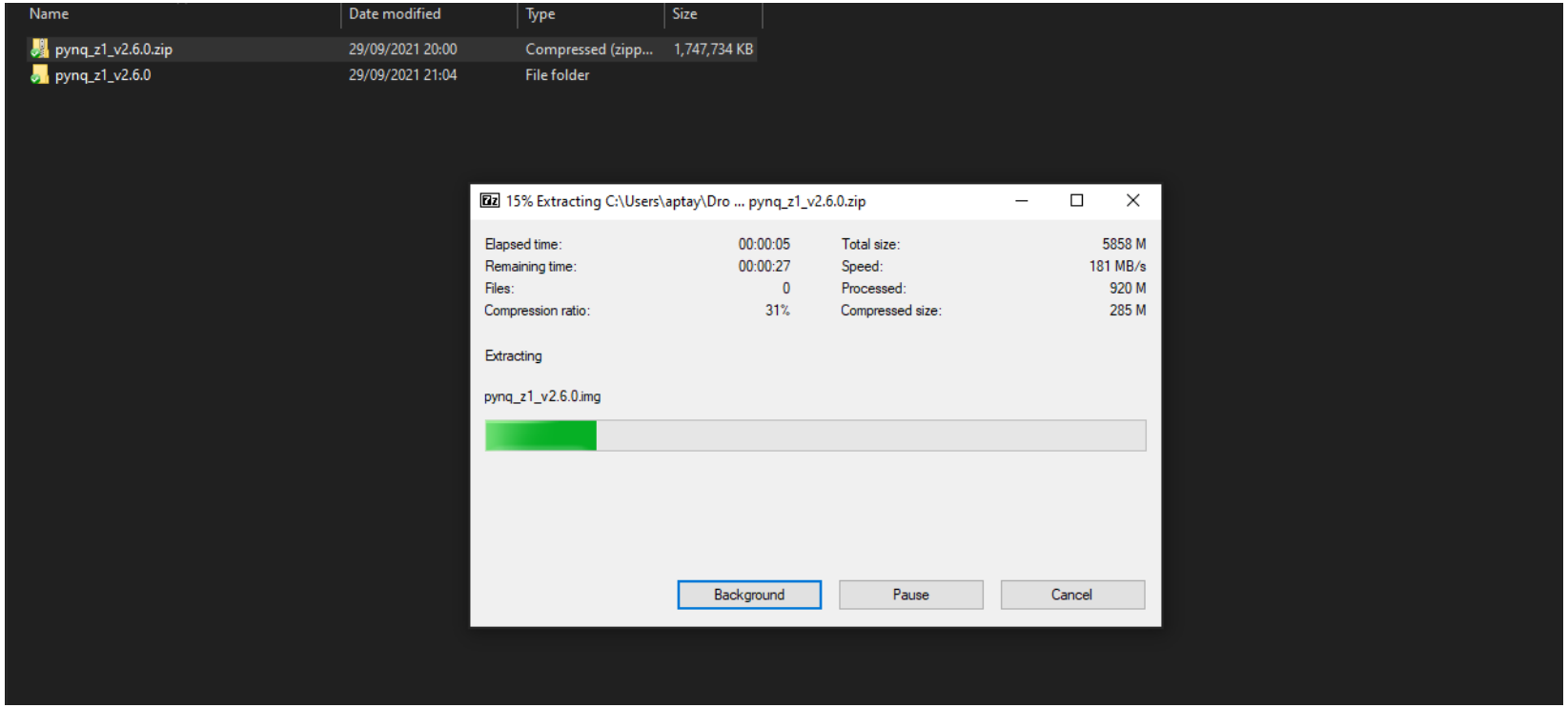
Lab : FFT Acceleration

Let Vivado manage the wrapper



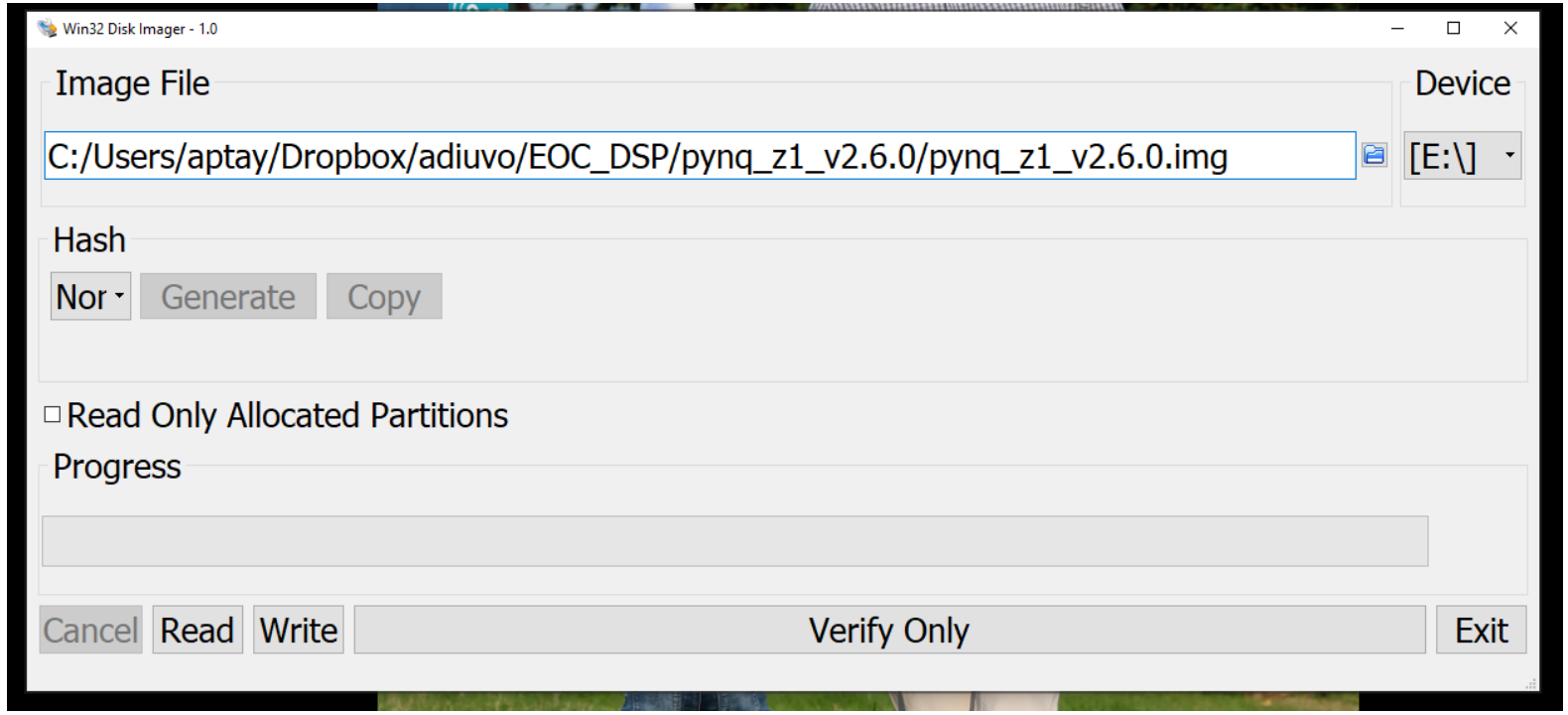
Lab : FFT Acceleration

Extract the downloaded PYNQ image



Lab : FFT Acceleration

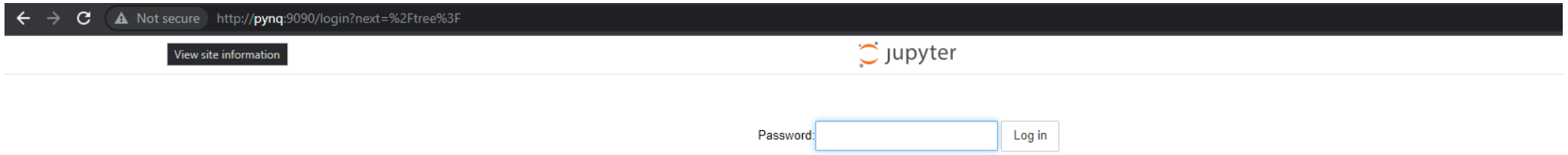
Write the image to a SD card. Once completed insert the SD card in the Arty Z7-20. Connect a Ethernet cable and power via a USB cable



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

Lab : FFT Acceleration

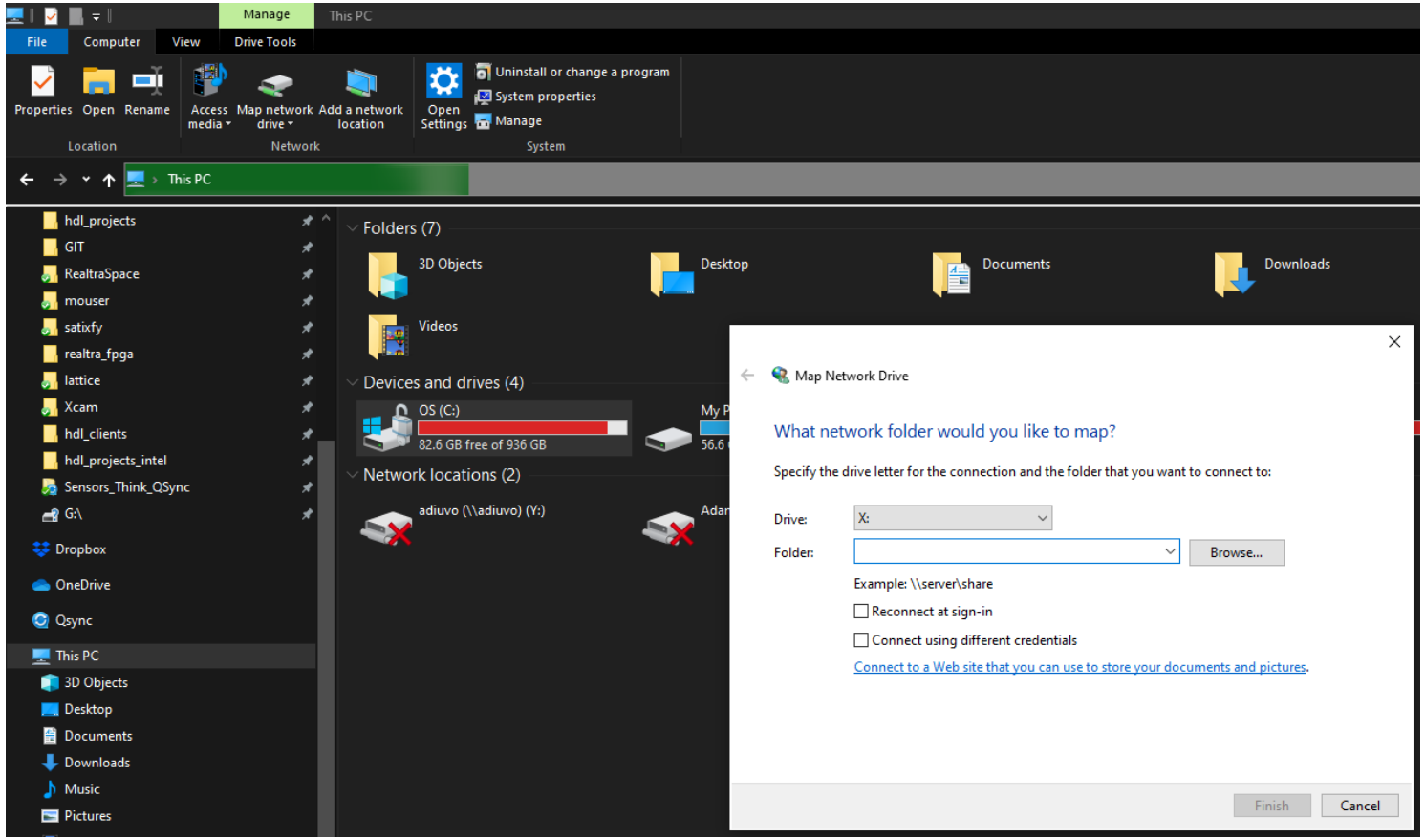
In a file explorer map a

network drive to

\\pynq \xilinx

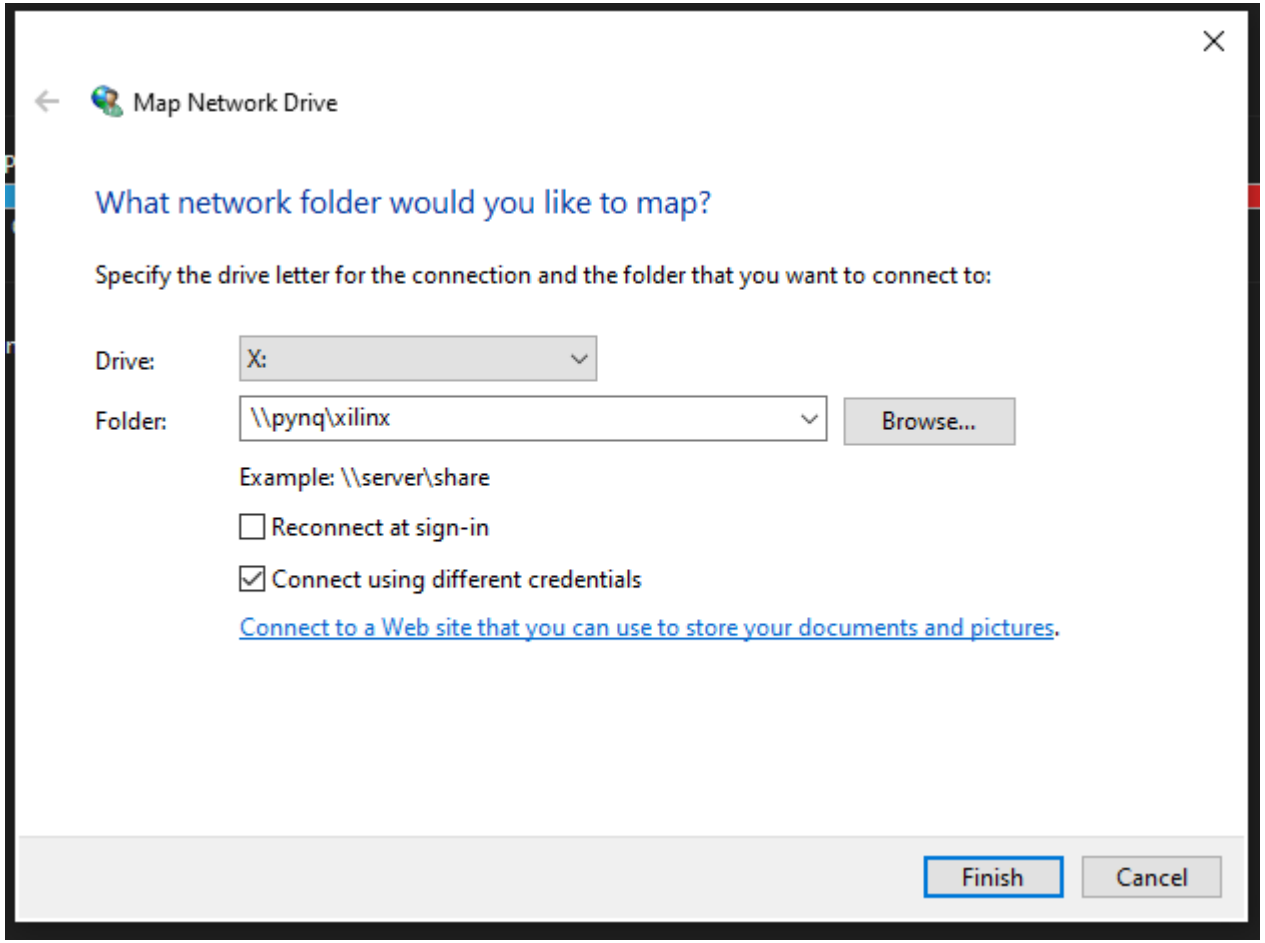
Select connect using different

credentials



Lab : FFT Acceleration

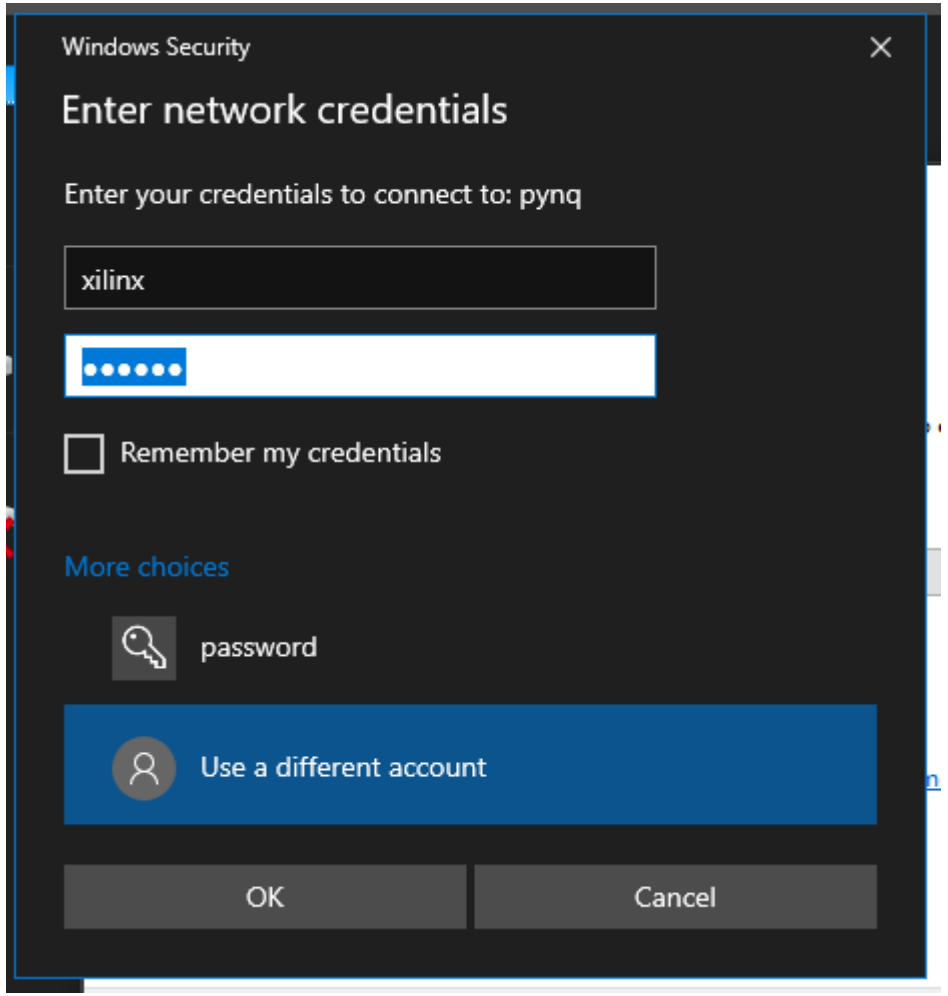
Completed Map drive, click OK



Lab : FFT Acceleration

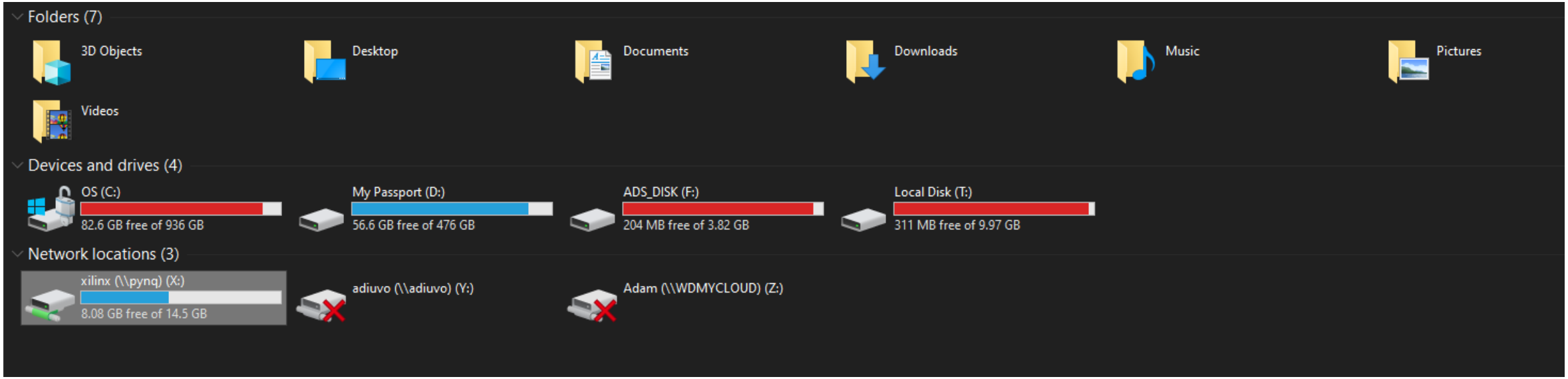
Enter the username and password as

Xilinx click OK



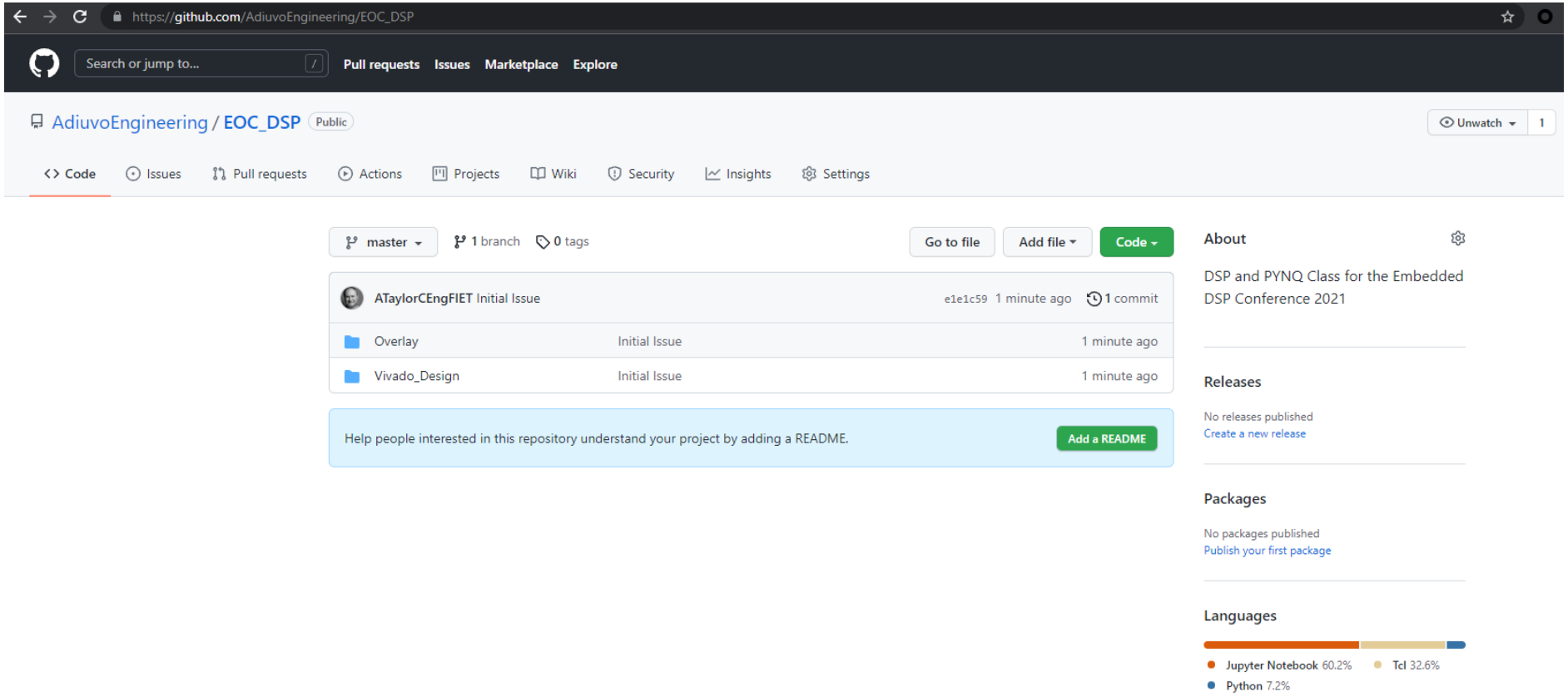
Lab : FFT Acceleration

The pynq drive should not appear as a samba server



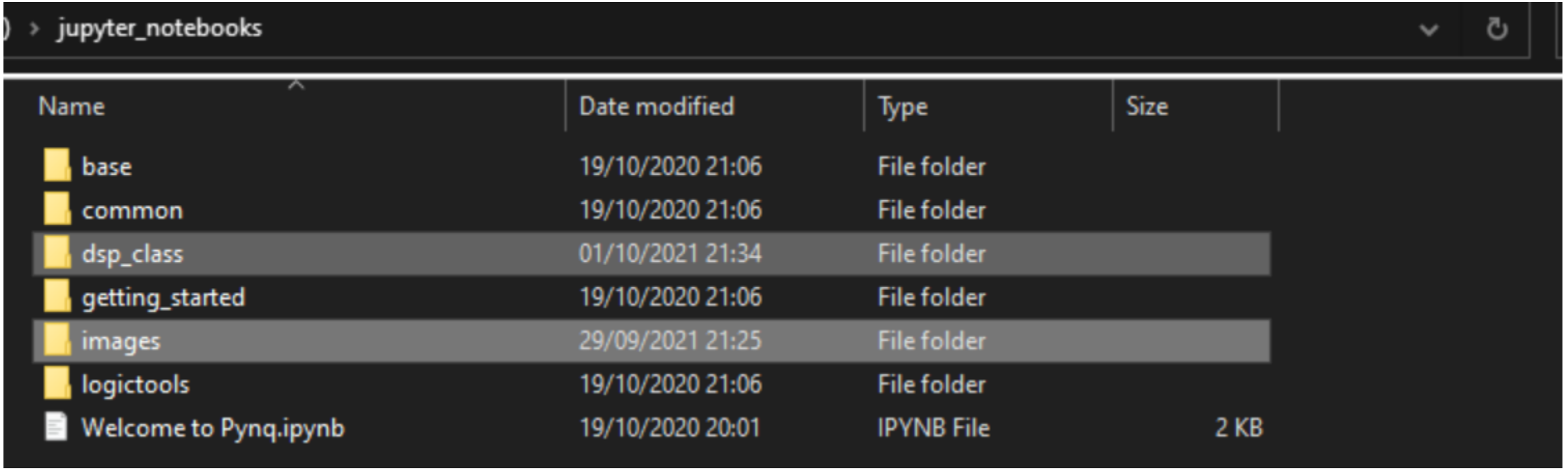
Lab : FFT Acceleration

Clone the repository - https://github.com/AdiuvoEngineering/EOC_DSP



Lab : FFT Acceleration

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

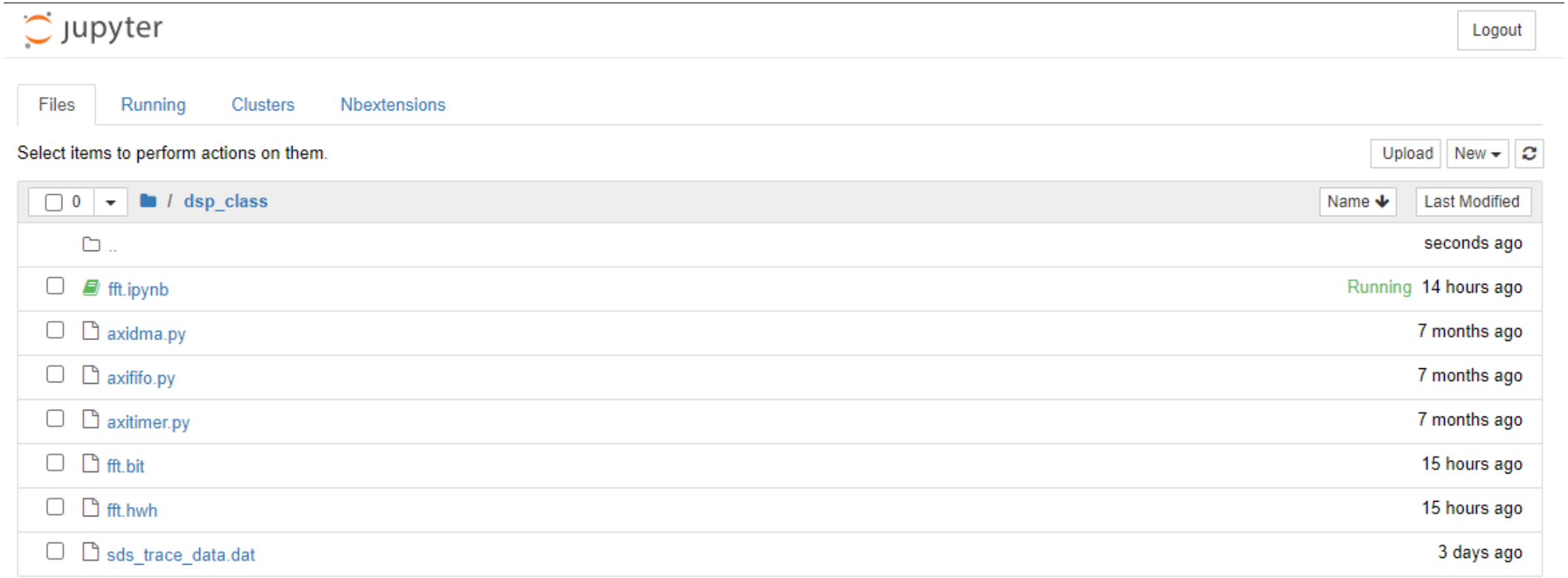


Lab : FFT Acceleration

You should see a new directory in the PYNQ environment, select DSP_CLASS

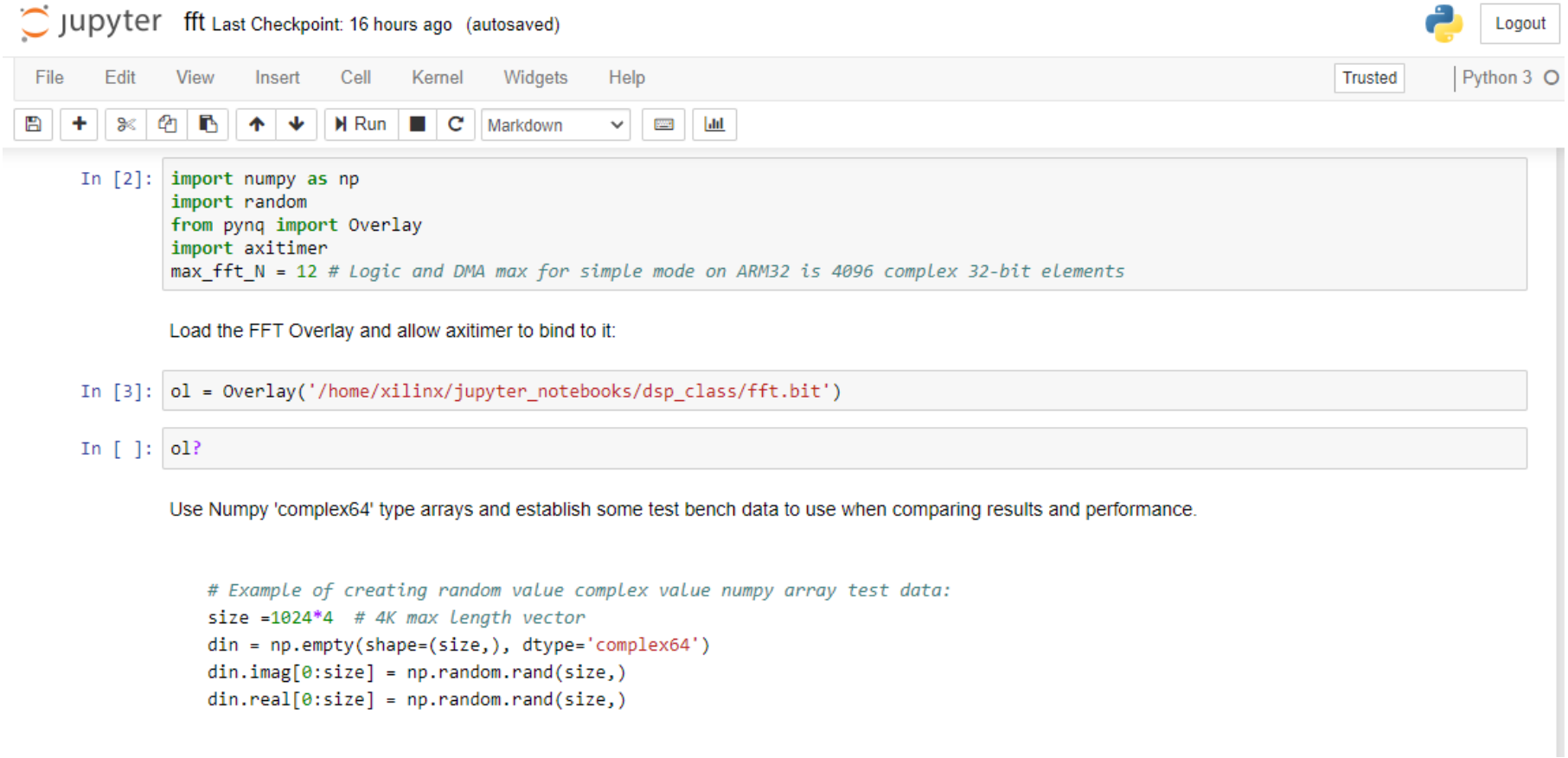
Lab : FFT Acceleration

Select fft.ipynb it will open and start running



Lab : FFT Acceleration

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



Questions?

ADIUVO
ENGINEERING AND TRAINING, LTD.

ADIUVO

ENGINEERING AND TRAINING, LTD.

www.adiuvoengineering.com

adam@adiuvoengineering.com