School and Conference on Edge Intelligence for Scientific Instrumentation and Reconfigurable Computing

Introduction to Laboratory Activities

Maynor G. Ballina

September 2026

UNIVERSITI MALAYSIA PERLIS

UNIVERSITI KEBANGSAAN MALAYSIA
The National University of Malaysia

The Abdus Salam International Centre for Theoretical Physics

UNIVERSITÀ DEGLI STUDI DI TRIESTE

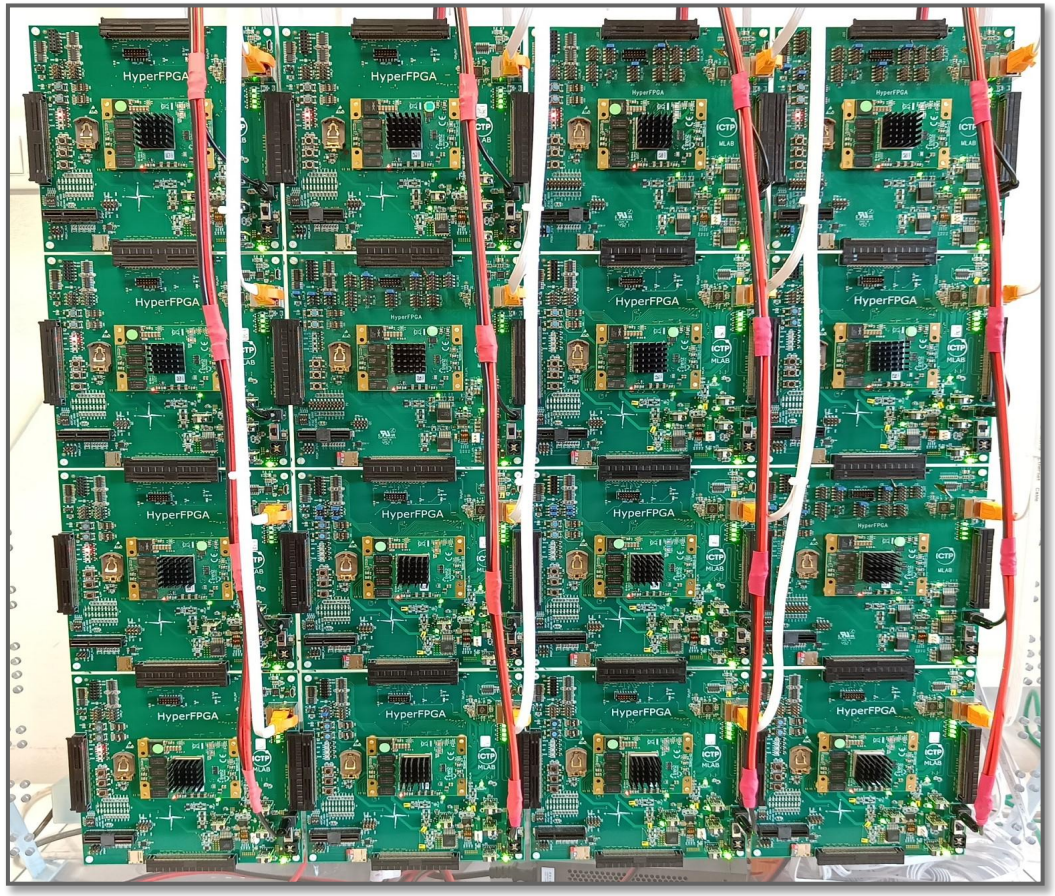
Welcome: Lab tutors

Dr. Romina Molina

Dr. Luis Garcia

Maynor Ballina

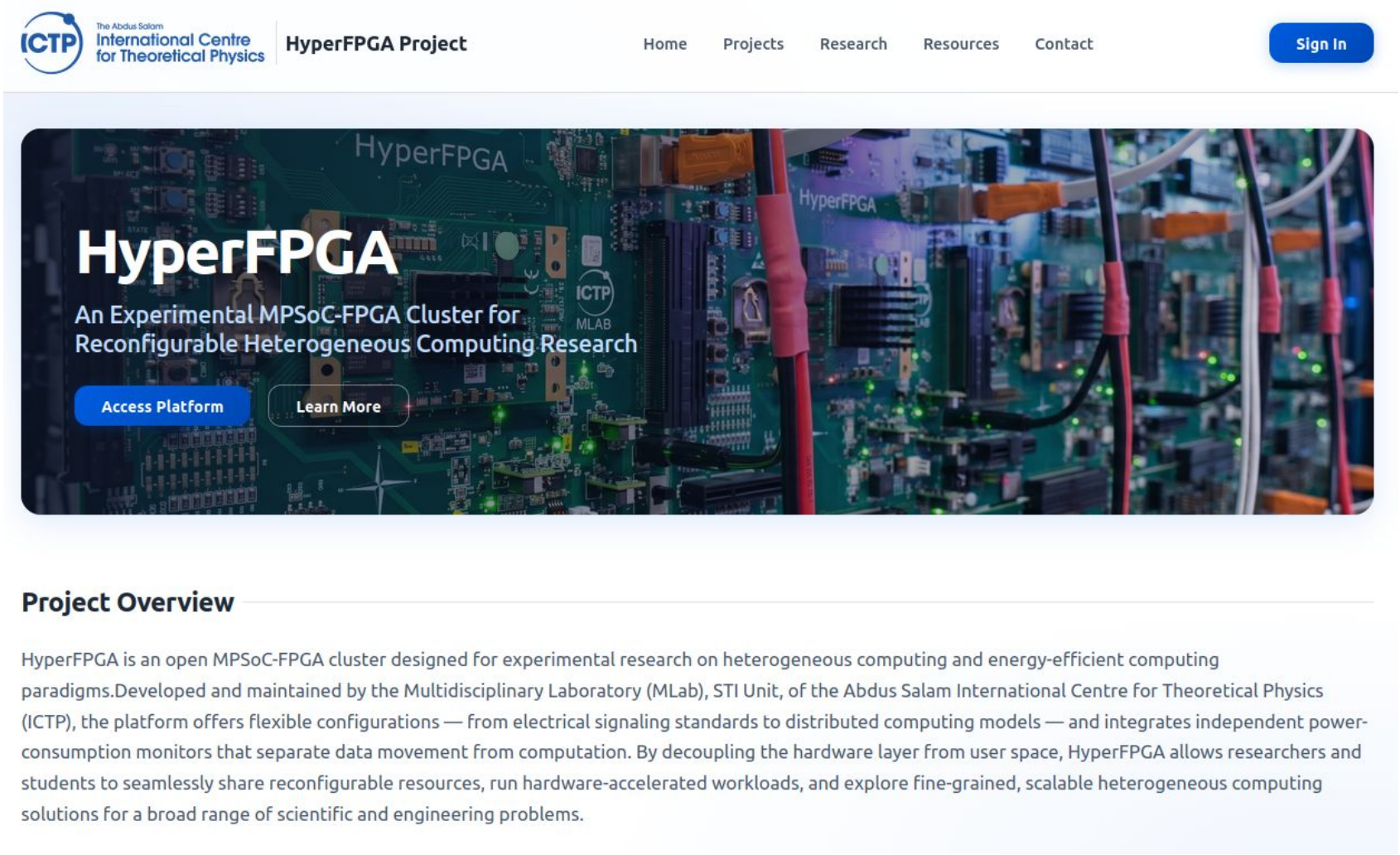
Kasun Mannatunga



Labs Overview:

- **Lab 0: User Environment and Access to the HyperFPGA Cluster**
- **Lab 1: Getting Started with MPSoC-FPGA and Comblock**
- **Lab 2 - Reconfigurable Instrumentation on MPSoC-FPGA**
- **Lab 3 - High-Level Synthesis (HLS) for Heterogeneous Computing**
- **Lab 4 - Training and Compression of Machine Learning Models**
- **Lab 5: End-to-End Workflow for Deploying Machine Learning Models on MPSoC-FPGA**
- **Final Project on Edge AI**

https://hyperfpga.sti.ictp.it



Resources

HyperFPGA BSP

Board support package for the HyperFPGA MPSoC-FPGA cluster nodes.

View Repository →

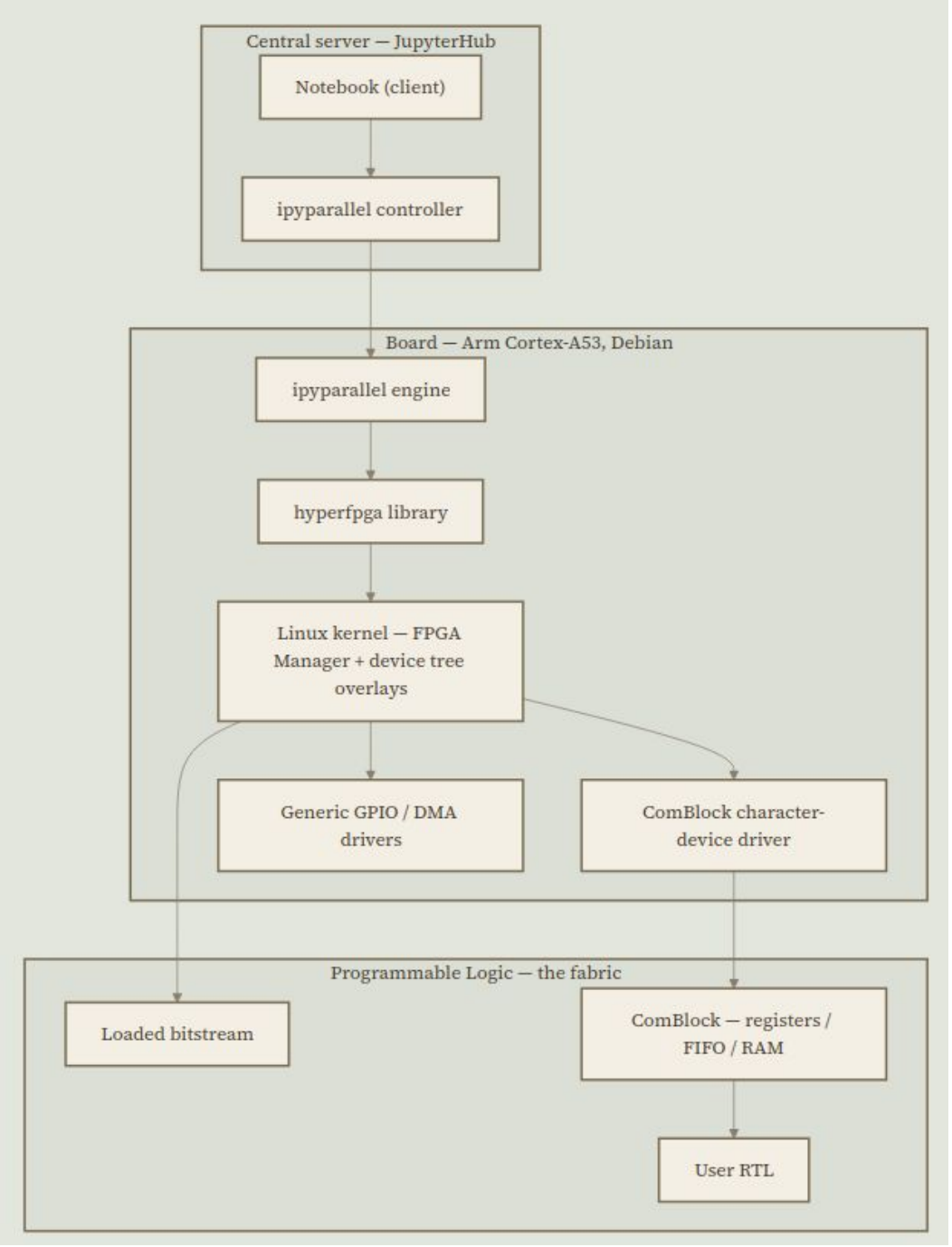
HyperFPGA User Guide

Comprehensive user guide covering the HyperFPGA comutils API, cluster usage, and hardware workflows.

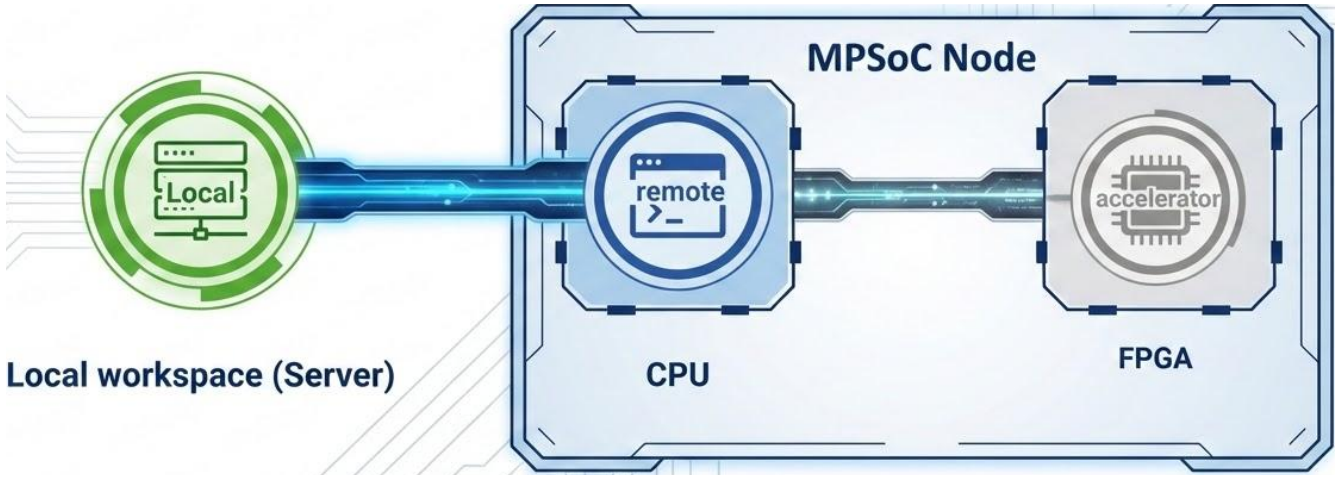
View User Guide →

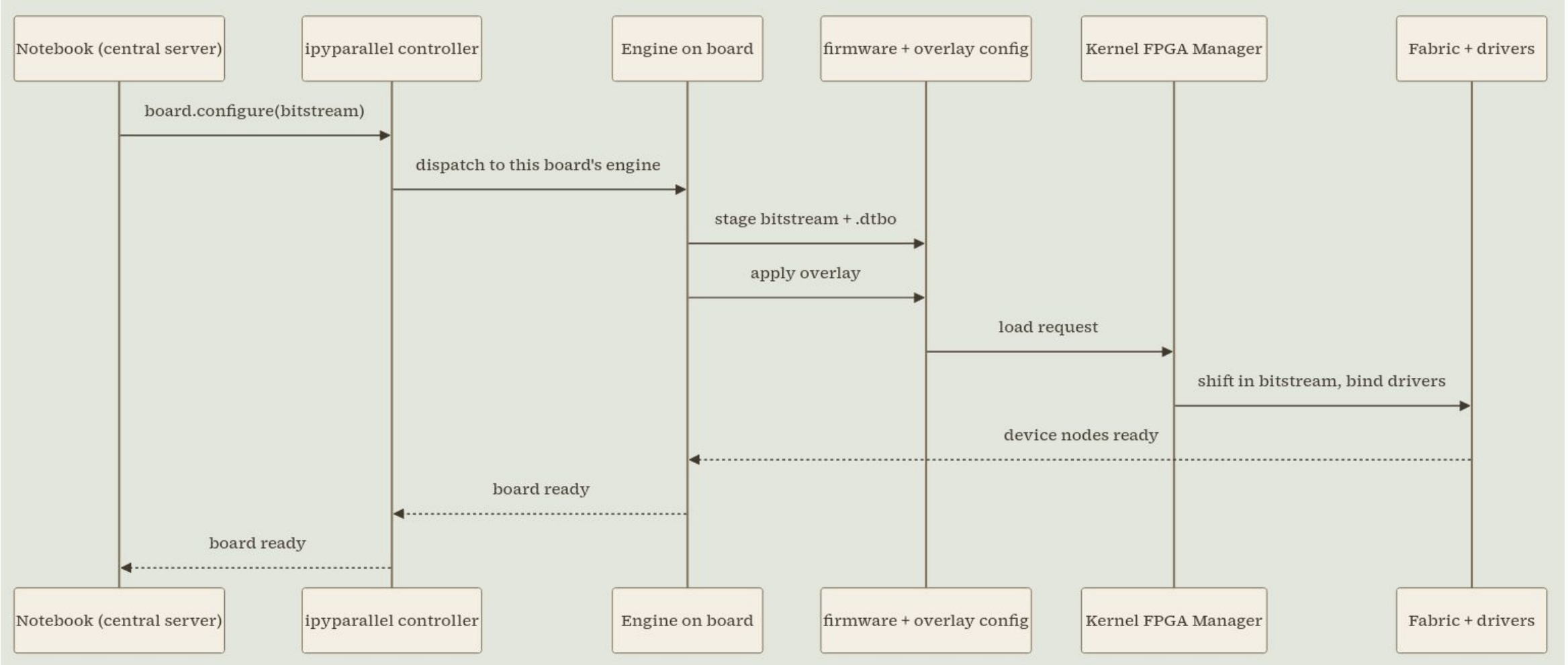
smr4200ux

smr4200ux



Your Heterogeneous Ecosystem




Notebook (central server)
ipyparallel controller
Engine on board
firmware + overlay config
Kernel FPGA Manager
Fabric + drivers
board.configure(bitstream)
dispatch to this board's engine
stage bitstream + .dtbo
apply overlay
load request
shift in bitstream, bind drivers
device nodes ready
board ready
board ready
Notebook (central server)
ipyparallel controller
Engine on board
firmware + overlay config
Kernel FPGA Manager
Fabric + drivers

School and Conference on Edge Intelligence for Scientific Instrumentation and Reconfigurable Computing

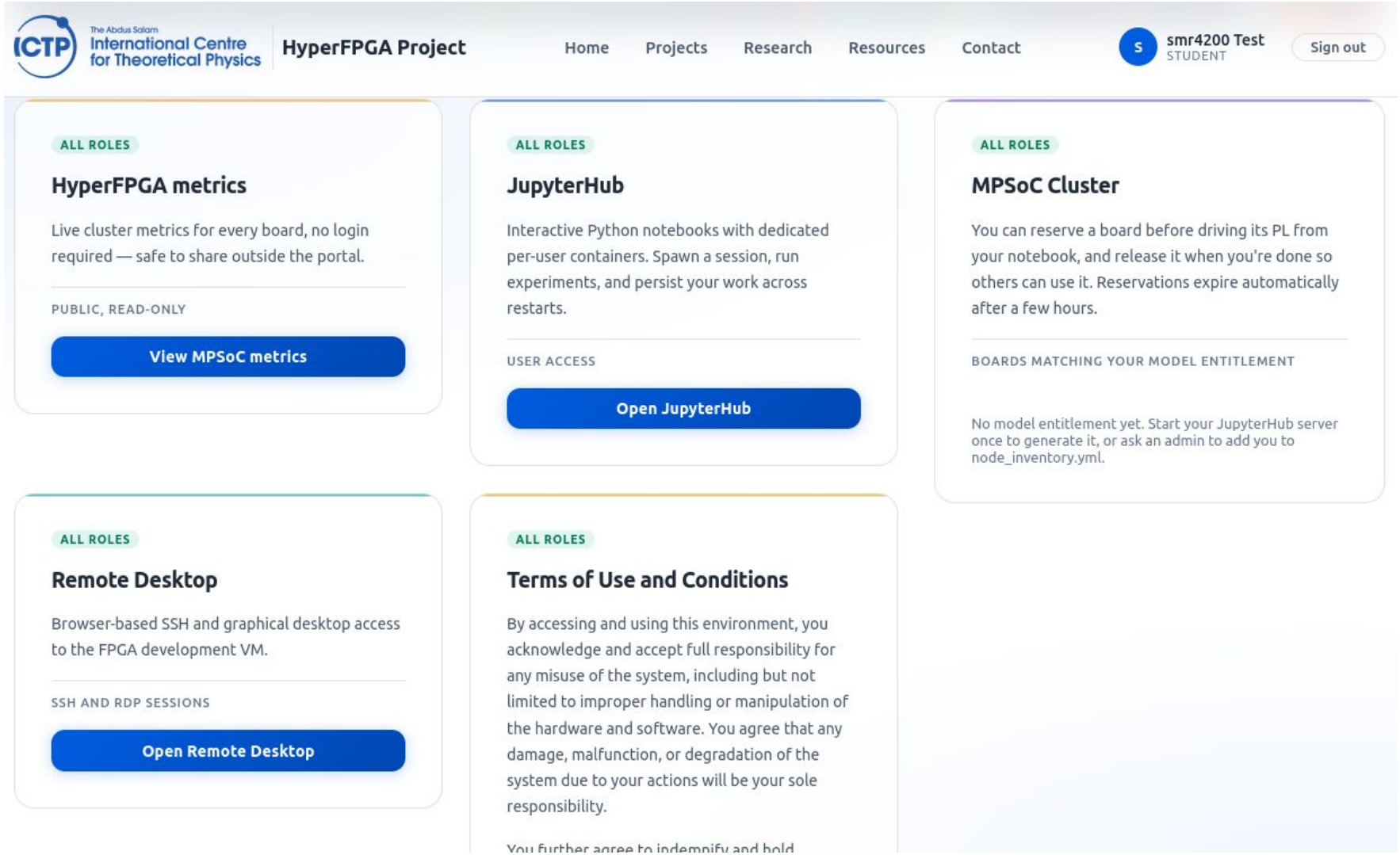
Lab 0: User Environment and Access to the HyperFPGA Cluster

Maynor G. Ballina

September 2026

UNIVERSITI MALAYSIA PERLIS
UniMAP
UNIVERSITI KEBANGSAAN MALAYSIA
The National University of Malaysia
ICTP The Abdus Salam International Centre for Theoretical Physics
UNIVERSITÀ DEGLI STUDI DI TRIESTE

User Dashboard

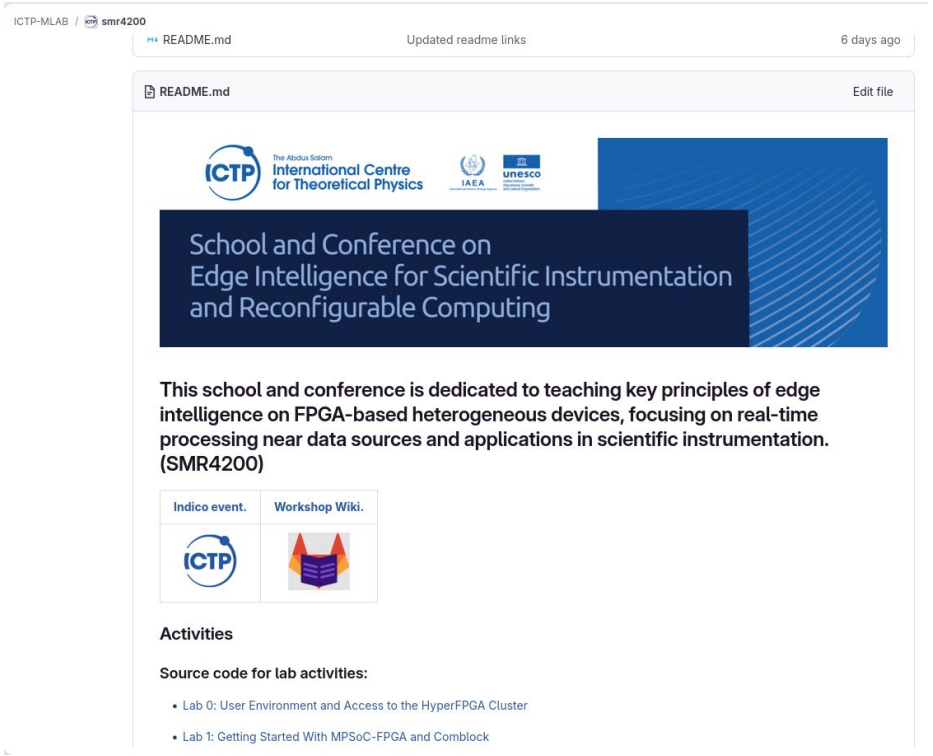


Virtual Machine

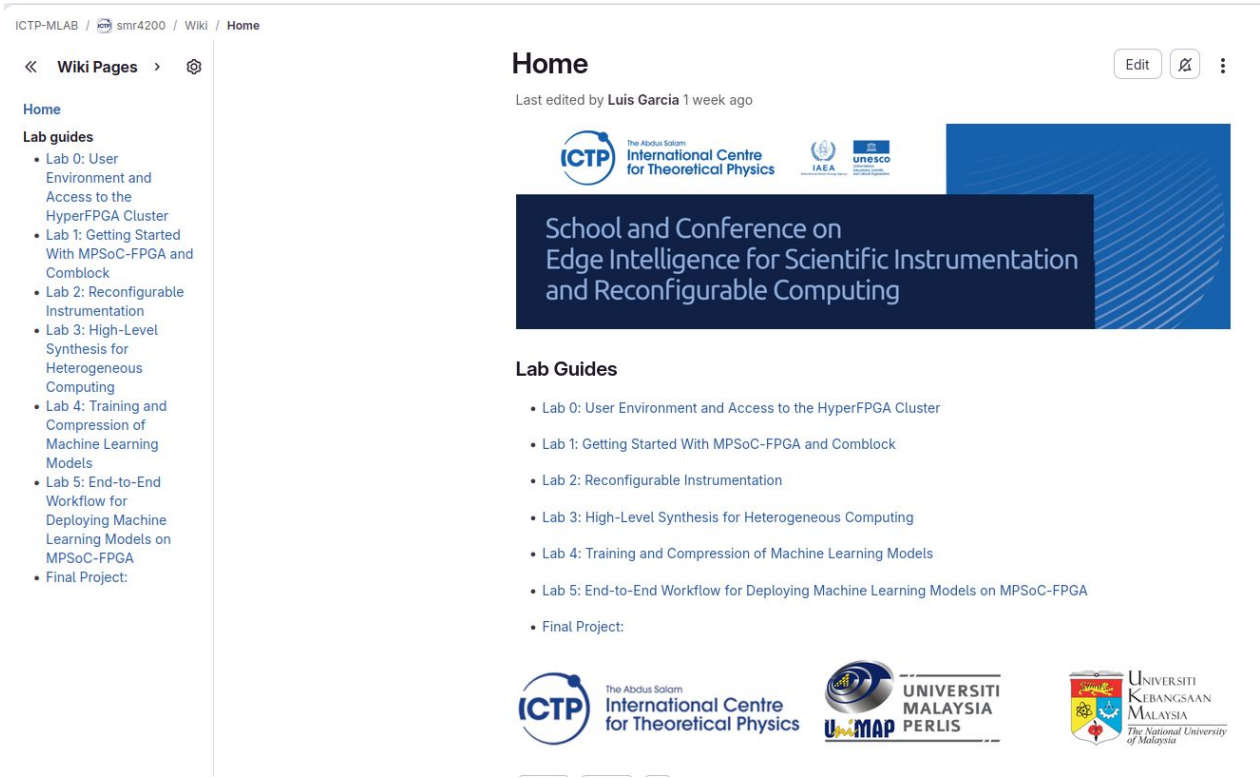
User: student
Password: workspace2026

Important Links

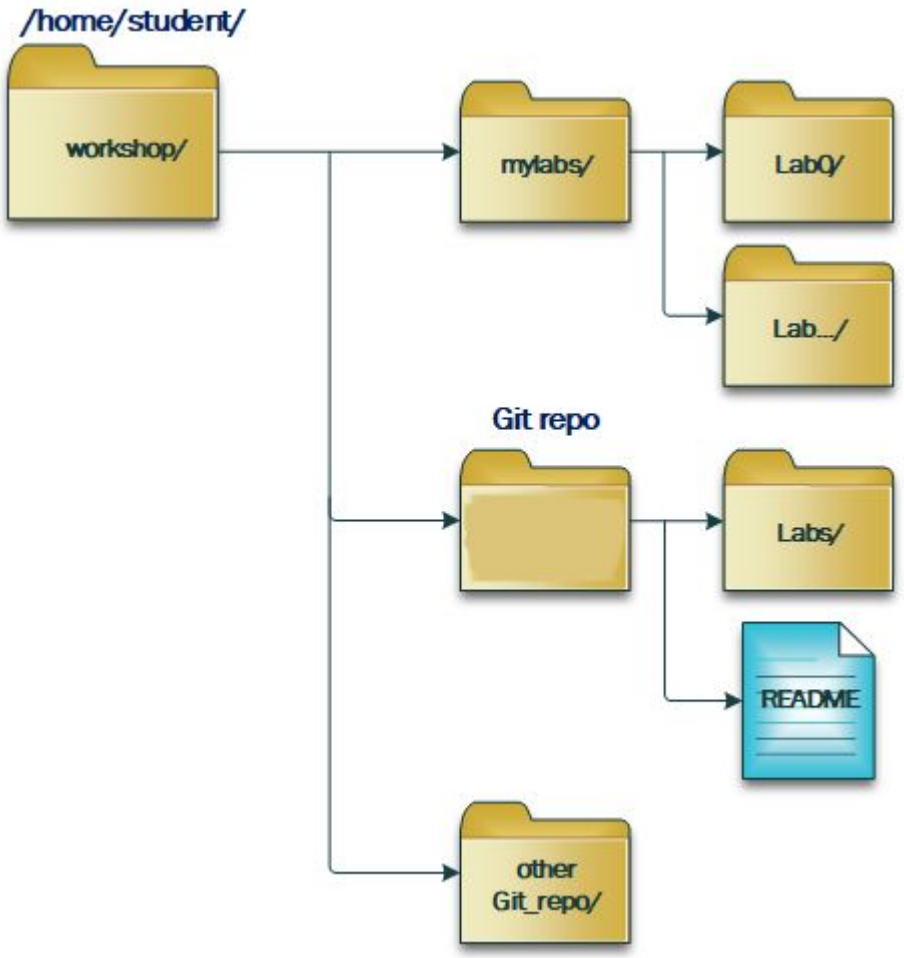
https://gitlab.com/ictp-mlab/smr-4200

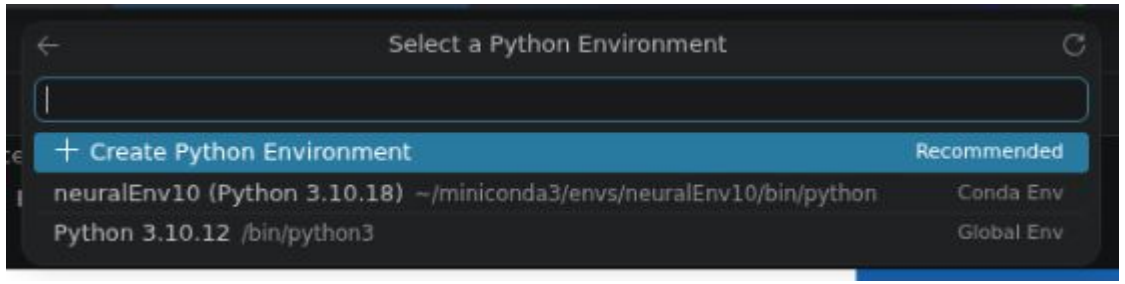


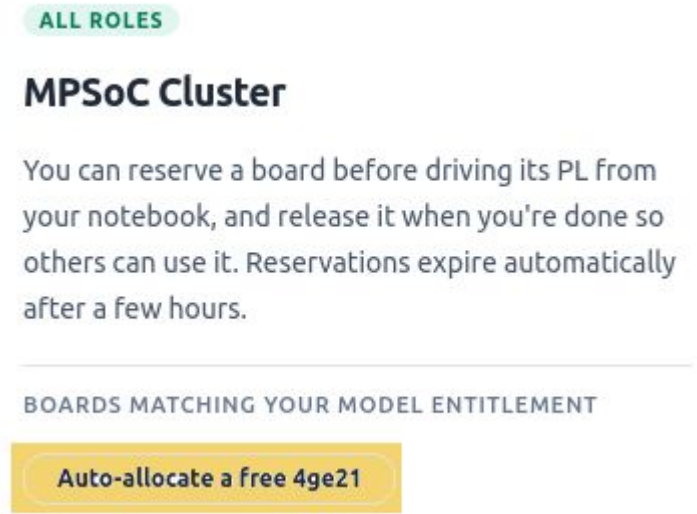
https://gitlab.com/ictp-mlab/smr-4200/-/wikis/home

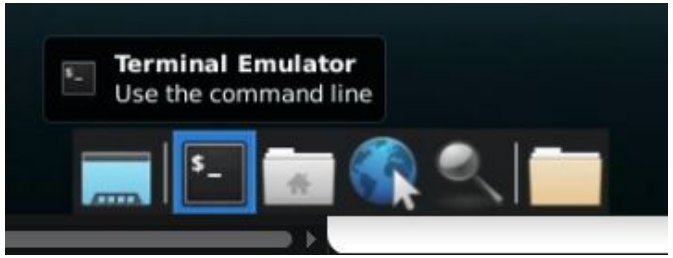


Objectives

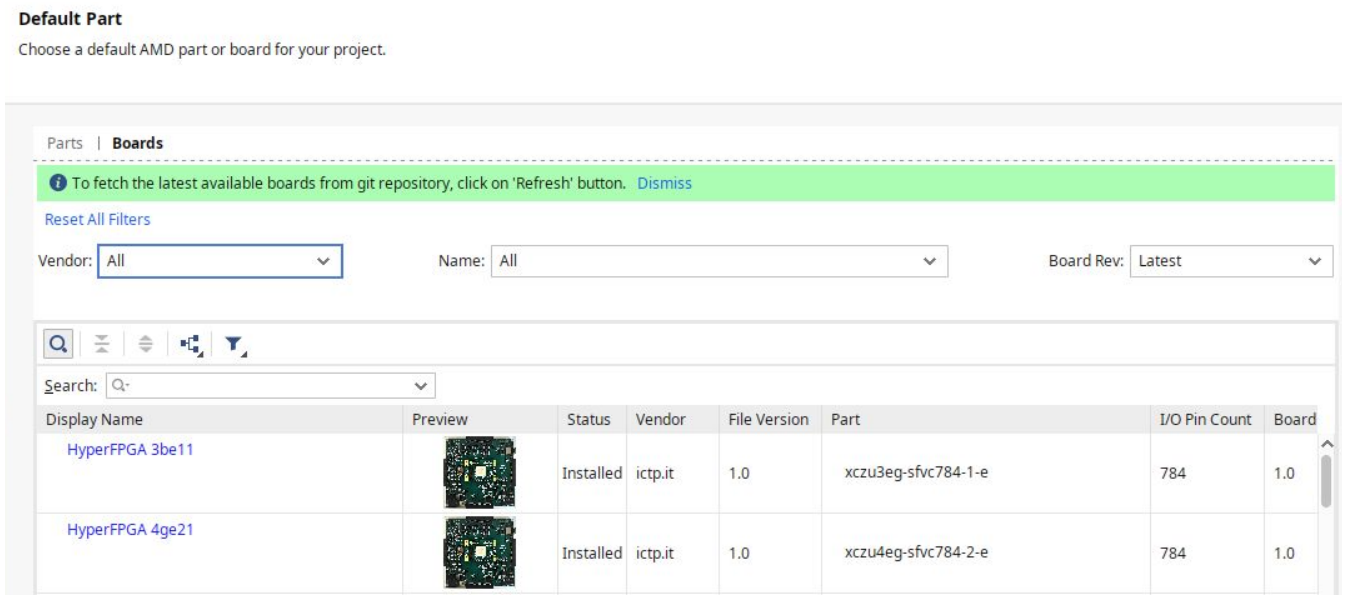

/home/student/
workshop/
mylabs/
Lab0/
Lab.../
Git repo
Labs/
README
other
Git_repo/

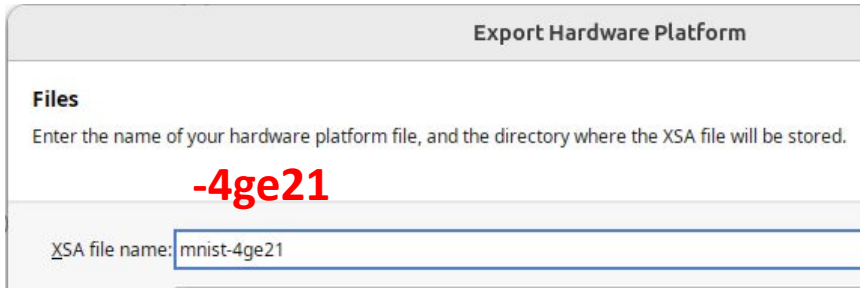

Select a Python Environment
Create Python Environment
Recommended
neuralEnv10 (Python 3.10.18) ~/miniconda3/envs/neuralEnv10/bin/python
Conda Env
Python 3.10.12 /bin/python3
Global Env

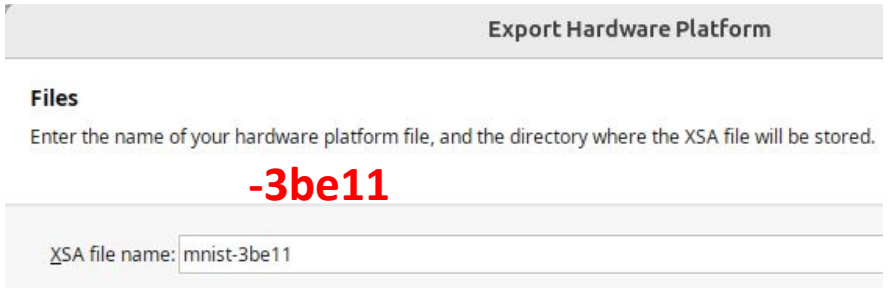

ALL ROLES
MPSoC Cluster
You can reserve a board before driving its PL from your notebook, and release it when you're done so others can use it. Reservations expire automatically after a few hours.
BOARDS MATCHING YOUR MODEL ENTITLEMENT
Auto-allocate a free 4ge21


Terminal Emulator
Use the command line

Development Workflow - Vivado







Workflow Overview: Connect, Program, Interact, Deprogram

Every session with a HyperFPGA node follows the same four stages, regardless of what your project does. This tutorial walks through each stage, one cell at a time:

1. **Connect**: hfc.available_models() asks hyperfpga-allocator which node(s) are available to you. request_nodes() then requests a node and assigns it to you for a set period of time. Finally, HyperFPGACluster(...) and cluster.create_profile() prepare the local ipyparallel/SSH profile needed to reach the node.
2. **Program**: await cluster.configure() reserves the node, uploads your bitstream (converting .xsa to .bit.bin/.dtbo via xsa2bin-builder if needed), and programs the FPGA with it. This step also prepares the board's CPU-side ipyparallel engine to run your code on the CPU or GPU, an .xsa file is not required to configure a node.
3. **Interact**: A Python kernel runs plain Python code on the board's ARM CPU cores.
4. **Deprogram**: When you're done, free the FPGA and release the node reservation using clean_cluster() and release_cluster(), so another user can use the node.

```
import hyperfpga_cluster as hfc
print(hfc.available_models())
test_node = await hfc.request_nodes('type') #4ge21, 3be11
cluster = hfc.HyperFPGACluster(nodes=test_node)
cluster.create_profile()
await cluster.configure()
```

```
await cluster.clean_cluster()
await cluster.release_cluster()
```

School and Conference on Edge Intelligence for Scientific Instrumentation and Reconfigurable Computing

Lab 1: Getting Started with MPSoC-FPGA and Comblock

Maynor G. Ballina

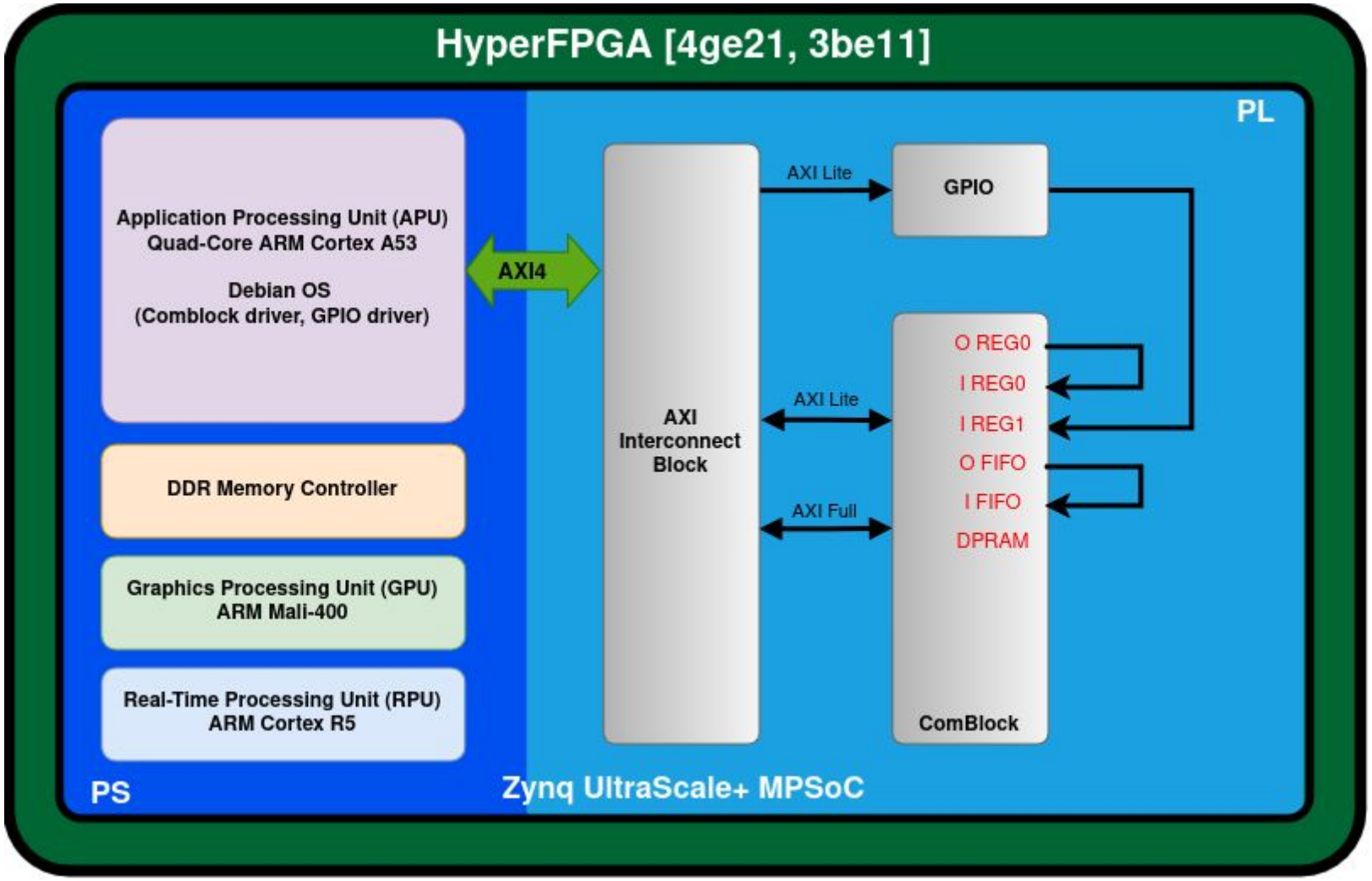
September 2026

UNIVERSITI MALAYSIA PERLIS
UniMAP
UNIVERSITI KEBANGSAAN MALAYSIA
The National University of Malaysia
ICTP The Abdus Salam International Centre for Theoretical Physics
UNIVERSITÀ DEGLI STUDI DI TRIESTE

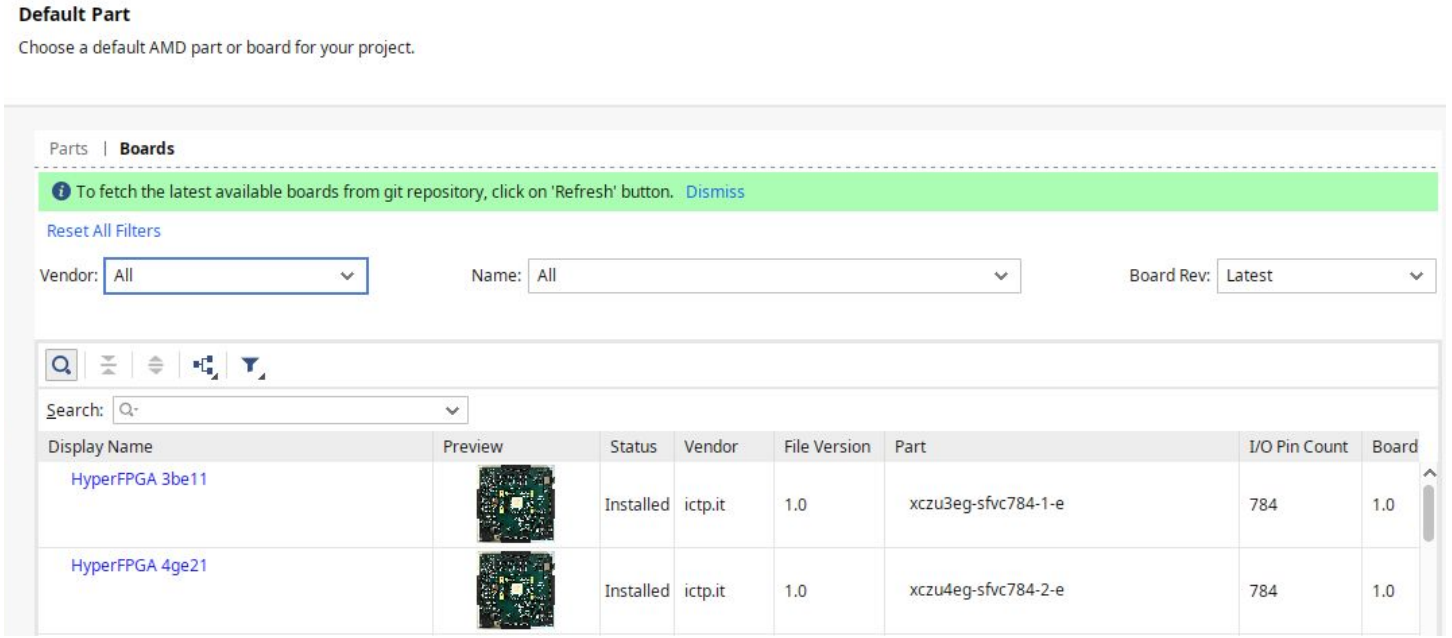
Objectives

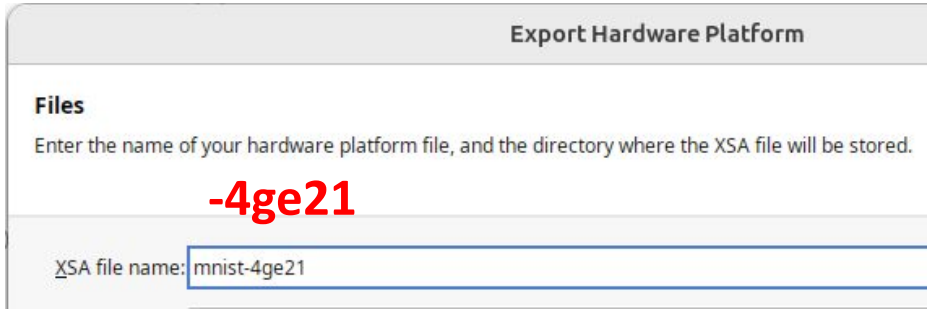
- Acquire knowledge of the MPSoC-FPGA design flow using the Vitis Unified Software Platform.
- Create the hardware to configure the Processing System (PS) and the Programmable Logic (PL) of the MPSoC, and understand the communication between the different components.
- Understand the usage and configuration of a HyperFPGA node.
- Develop a 'Python' application (in Jupyter Notebook) that will run on the PS.
- Test the complete design on the HyperFPGA platform to verify functionality using Jupyter Hub.

Design description



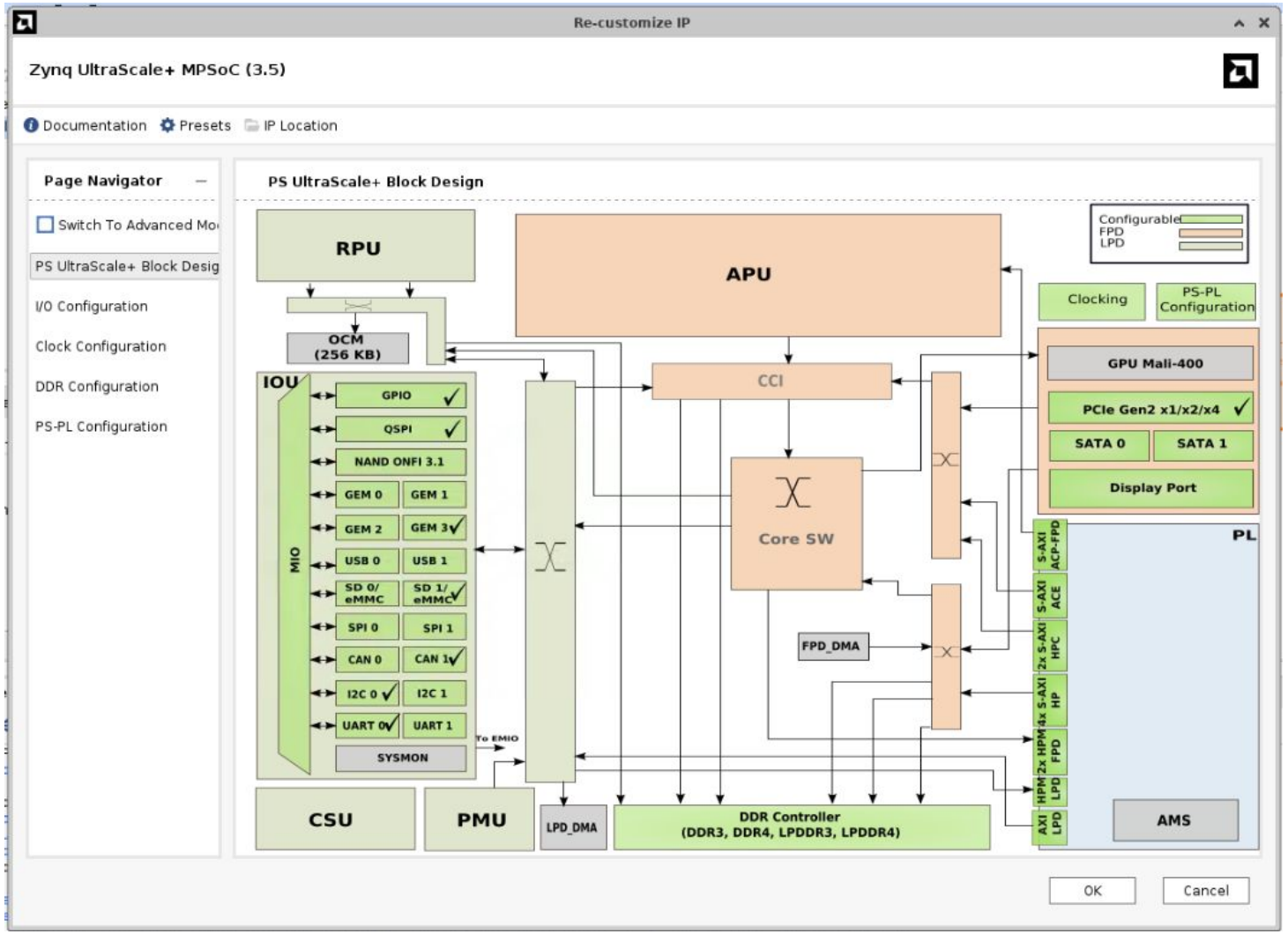
Development Workflow - Vivado



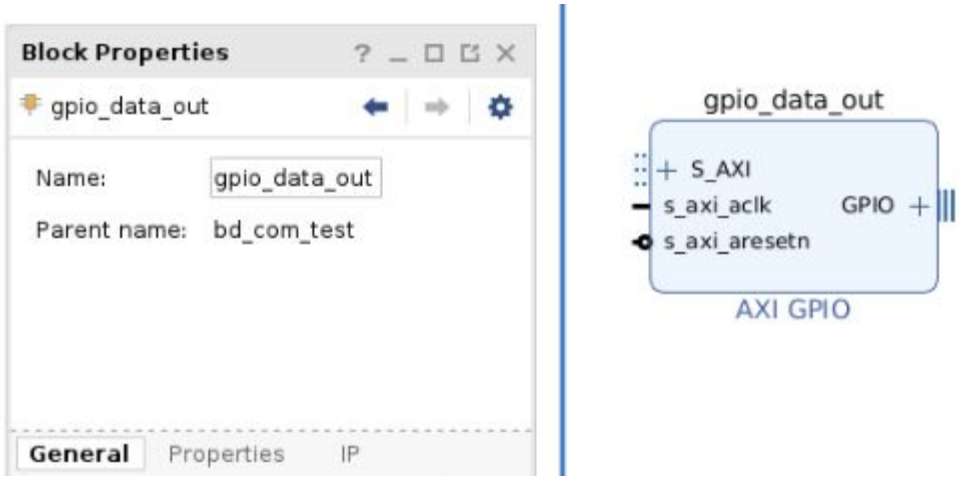


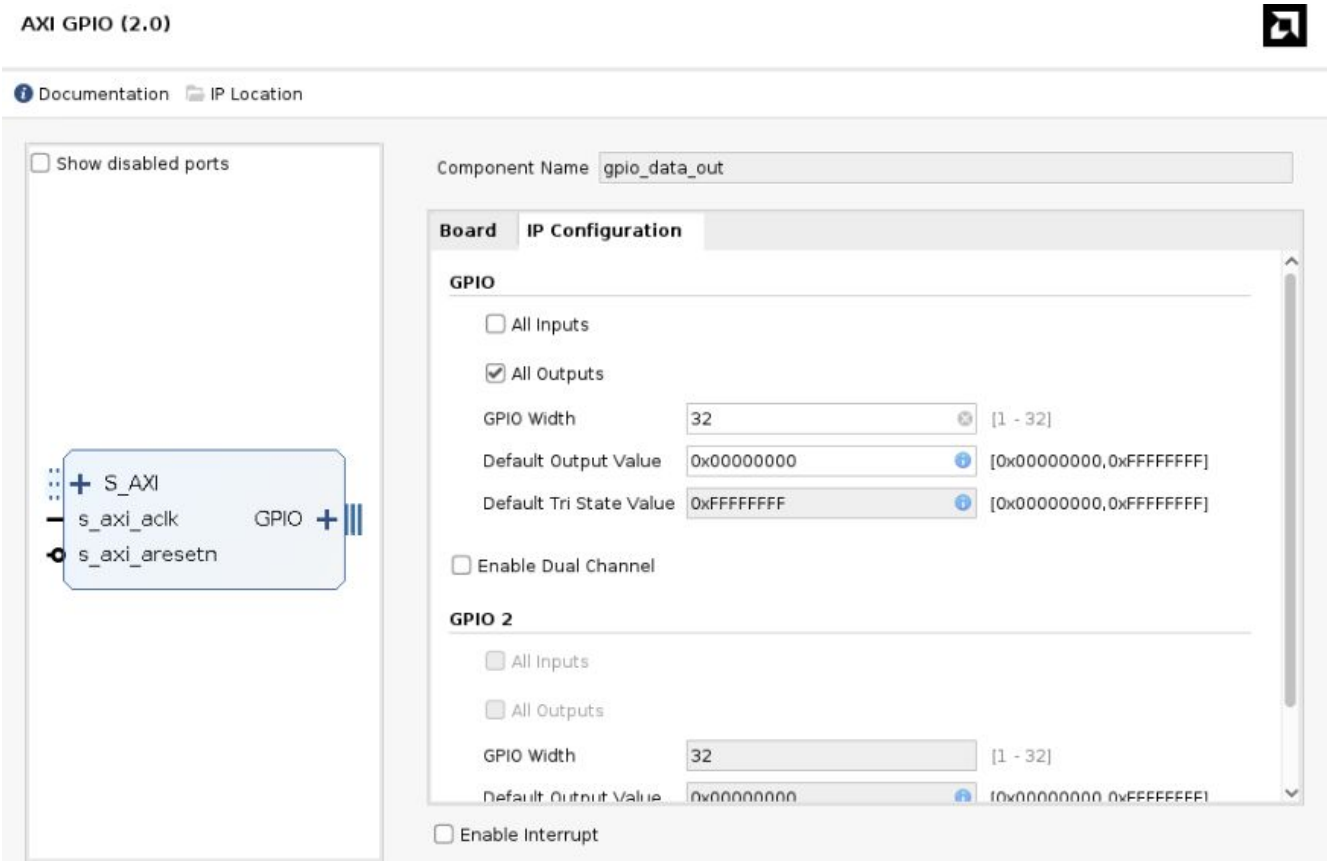
Hardware


zynq_ultra_ps_e_0
maxihpm0_lpd_aclk
M_AXI_HPM0_LPD
pl_resetn0
pl_clk0
ZYNQ
UltraSCALE+
Zynq UltraScale+ MPSoC


Re-customize IP
Zynq UltraScale+ MPSoC (3.5)
Documentation
Presets
IP Location
Page Navigator
Switch To Advanced Mo
PS UltraScale+ Block Desig
I/O Configuration
Clock Configuration
DDR Configuration
PS-PL Configuration
PS UltraScale+ Block Design
RPU
APU
OCM (256 KB)
CCI
Core SW
IOU
MIO
GPIO
QSPI
NAND ONFI 3.1
GEM 0
GEM 1
GEM 2
GEM 3
USB 0
USB 1
SD 0/ eMMC
SD 1/ eMMC
SPI 0
SPI 1
CAN 0
CAN 1
I2C 0
I2C 1
UART 0
UART 1
SYSMON
To EMIO
FPD_DMA
LPD_DMA
CSU
PMU
DDR Controller (DDR3, DDR4, LPDDR3, LPDDR4)
Configurable
FPD
LPD
Clocking
PS-PL Configuration
GPU Mali-400
PCIe Gen2 x1/x2/x4
SATA 0
SATA 1
Display Port
PL
S-AXI ACP-FPD
S-AXI ACE
2x S-AXI HPC
4x S-AXI HP
2x HPM FPD
HPM LPD
AXI LPD
AMS
OK
Cancel

Steps


Block Properties
gpio_data_out
Name:
gpio_data_out
Parent name:
bd_com_test
General
Properties
IP
gpio_data_out
S_AXI
s_axi_aclk
s_axi_aresetn
GPIO
AXI GPIO


AXI GPIO (2.0)
Documentation
IP Location
Show disabled ports
S_AXI
s_axi_aclk
s_axi_aresetn
GPIO
Component Name
gpio_data_out
Board
IP Configuration
GPIO
All Inputs
All Outputs
GPIO Width
32
[1 - 32]
Default Output Value
0x00000000
[0x00000000,0xFFFFFFFF]
Default Tri State Value
0xFFFFFFFF
[0x00000000,0xFFFFFFFF]
Enable Dual Channel
GPIO 2
All Inputs
All Outputs
GPIO Width
32
[1 - 32]
Enable Interrupt

Steps

I/O Registers

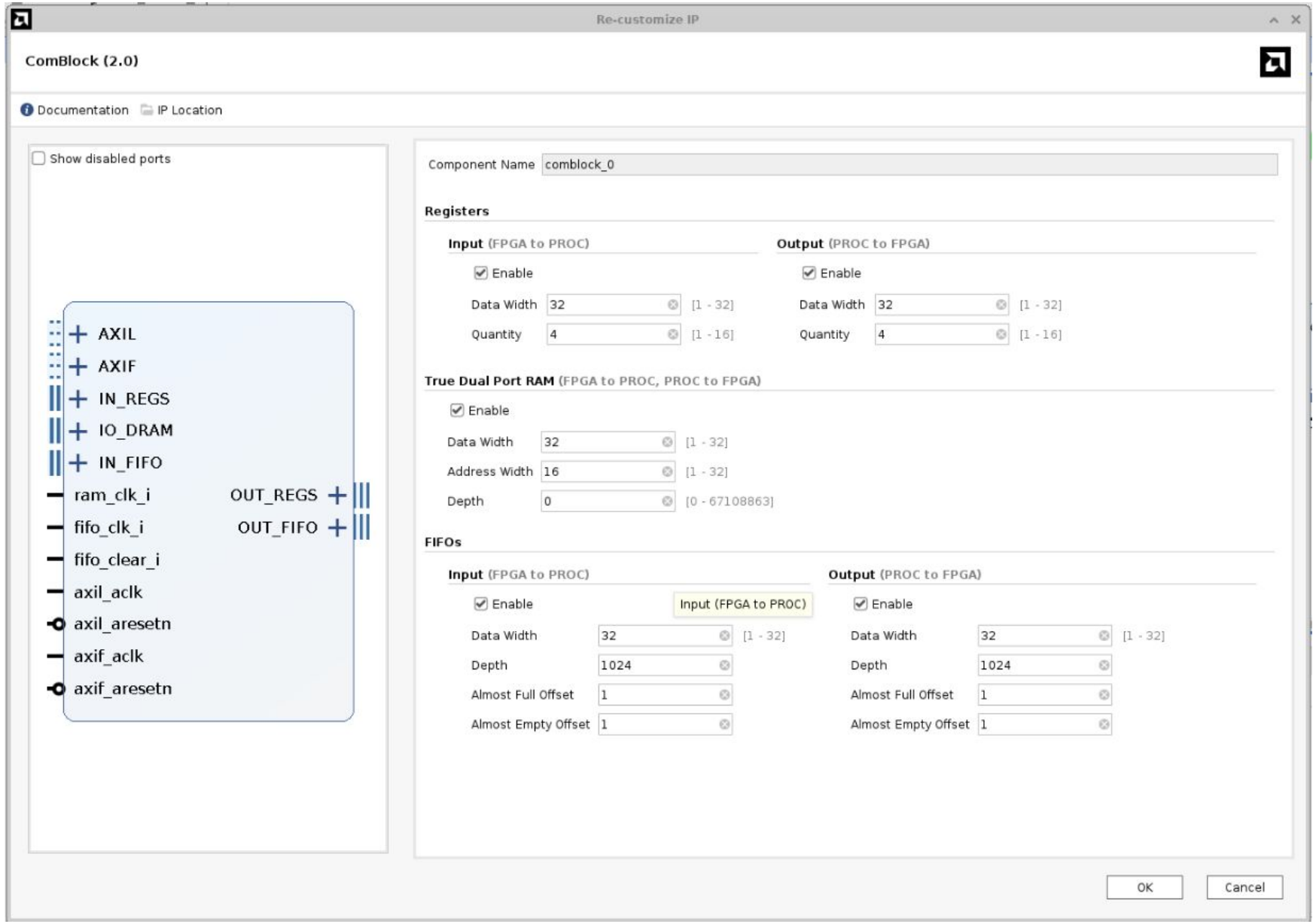
- 32 Bits

I/O FIFO

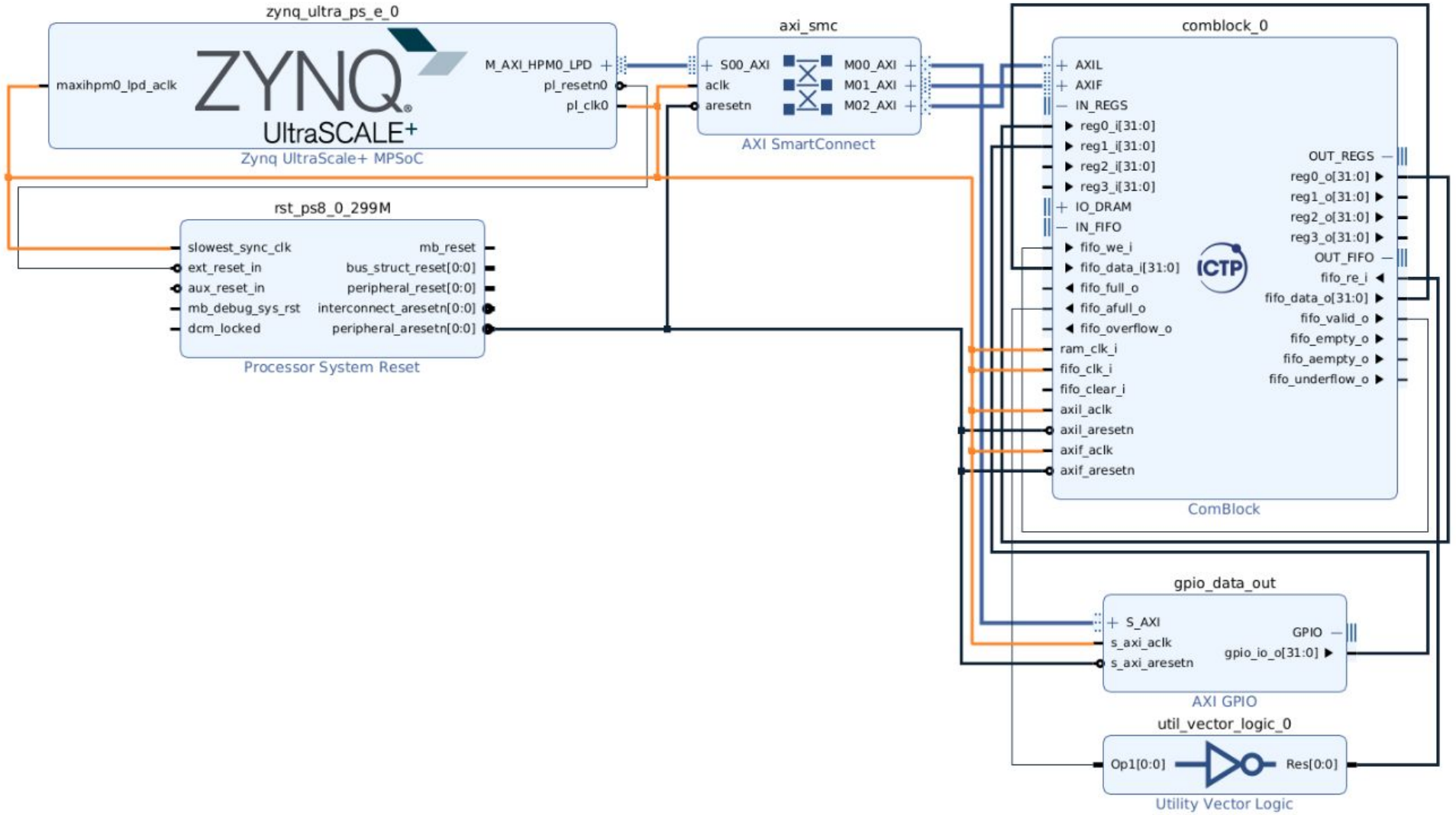
- 32 Bits
- 1024 Depth

RAM

- 32 bits
- 16 Address width

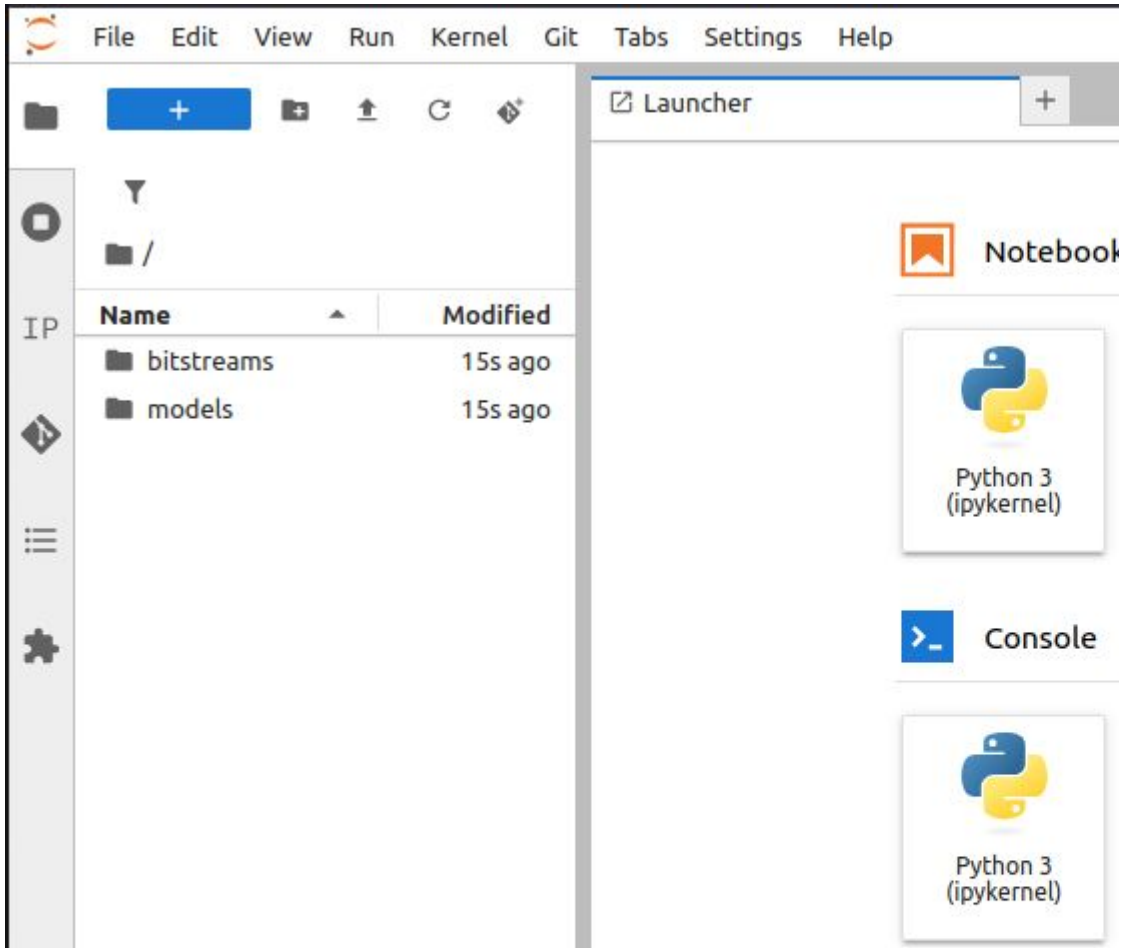


Final Block Design



Jupyter Hub

- Upload .xsa file to bitstreams folder
- Create smr4200 folder
- Upload the All the files inside the Jupyterhub folder from the git repo
- Create your code based on lab0
- Create your interaction with the hardware
- Validate your design



Thank you

Q&A

mballina@ictp.it