# Soc-FPGA Development Framework

1st Mesoamerican Workshop on Reconfigurable X-ray Scientific Instrumentation for Cultural Heritage

**Luis Guillermo García Ordóñez**

IAEA International Atomic Energy Agency
*Atoms for Peace and Development*

IUPAP Centenary YEARS 1922 - 2022

Gobierno de la República Guatemala
Secretaría Nacional de Ciencia y Tecnología

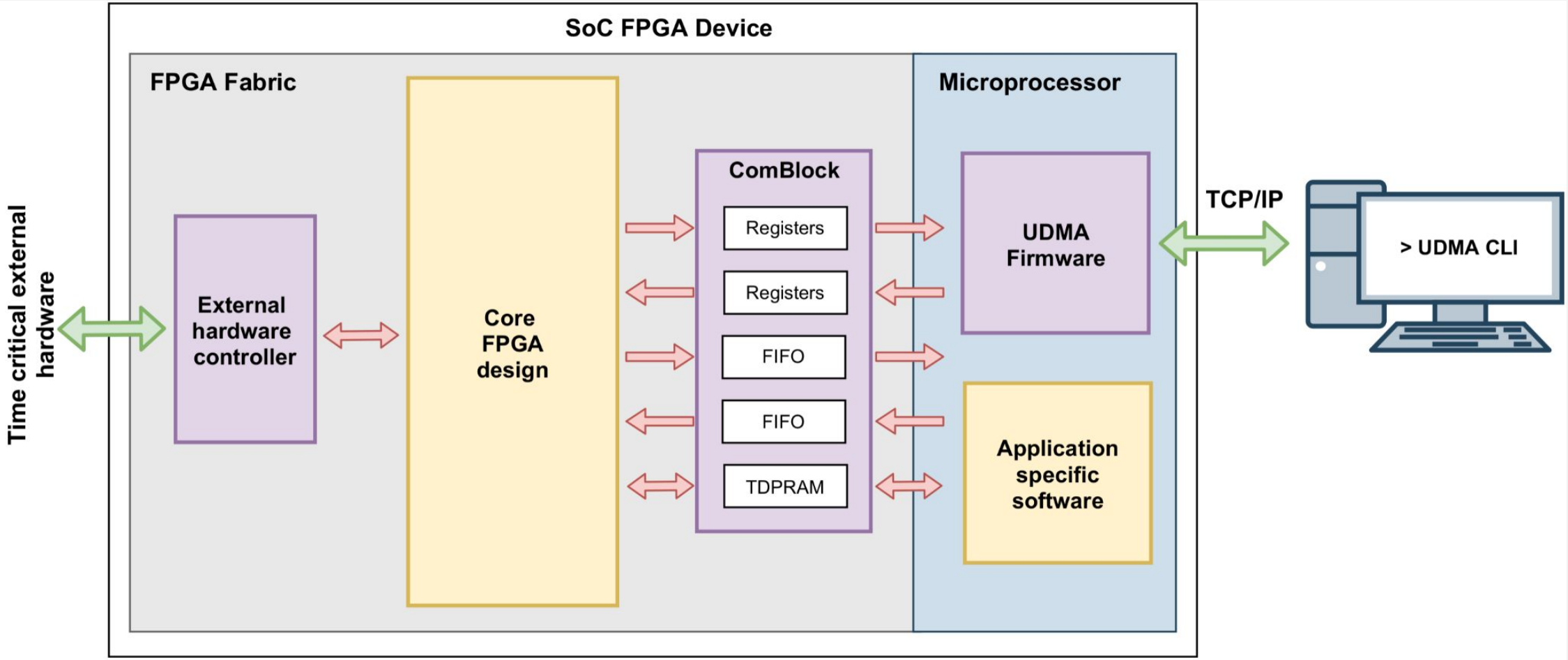
Carolina Academia Coactemalensis Inter Cæteras Orbis Conspicua

ICTP The Abdus Salam International Centre for Theoretical Physics

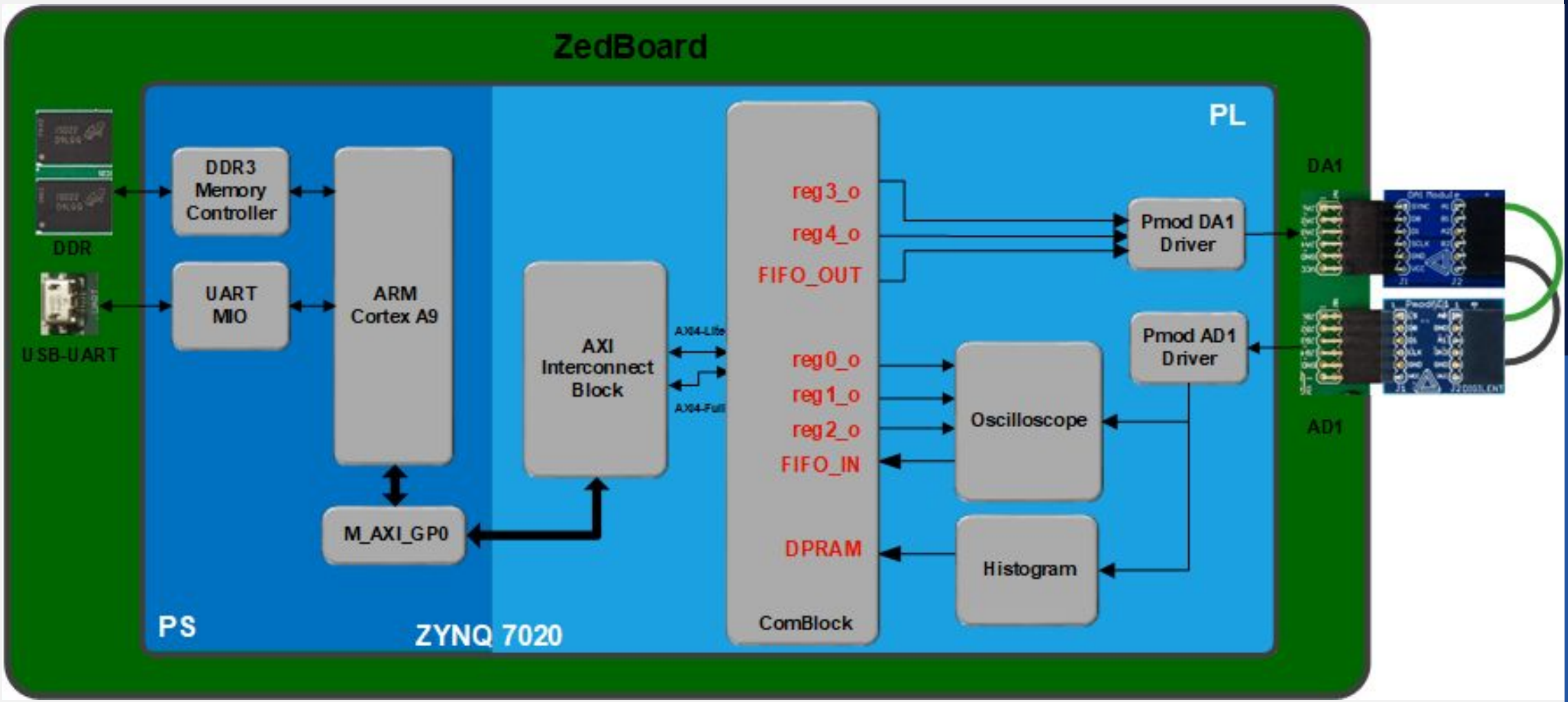
# Objectives

- Use the ComBlock in the simple mode for data transfer from the PC.
- Transfer data between the PC and the FPGA through the UDMA and the ComBlock.
- Interact within JupyterLab with custom hardware.
- Familiarize with the usage of the FreeRTOS lwIP stack.
- Learn how to work in a collaborative environment with version control tools, employing **tcl** commands for hardware recreation.
- Understand the benefits of a higher abstraction level when using the UDMA environment for interaction.
- Implement a digital oscilloscope and a histogram counter on the SoC-FPGA.

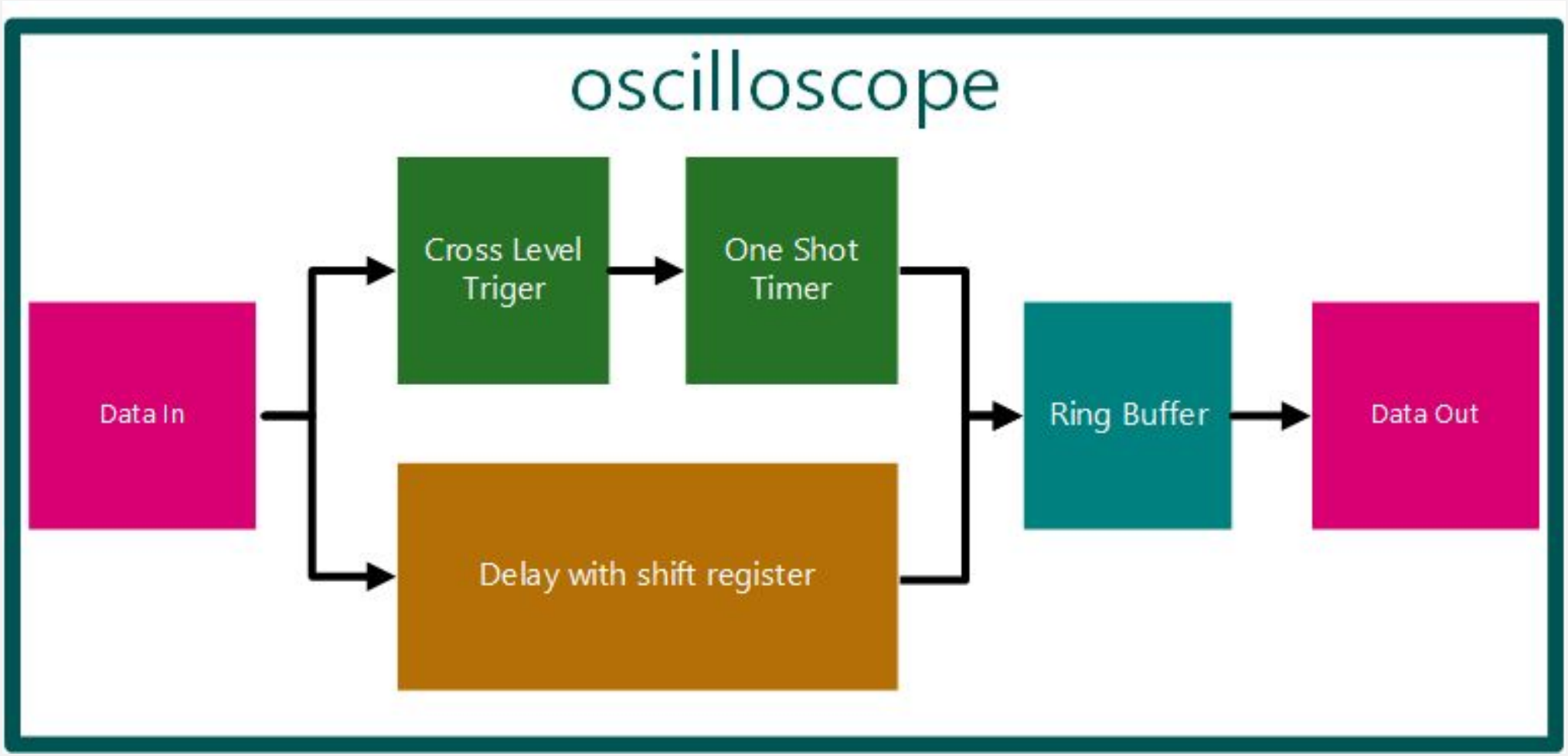
# Our framework



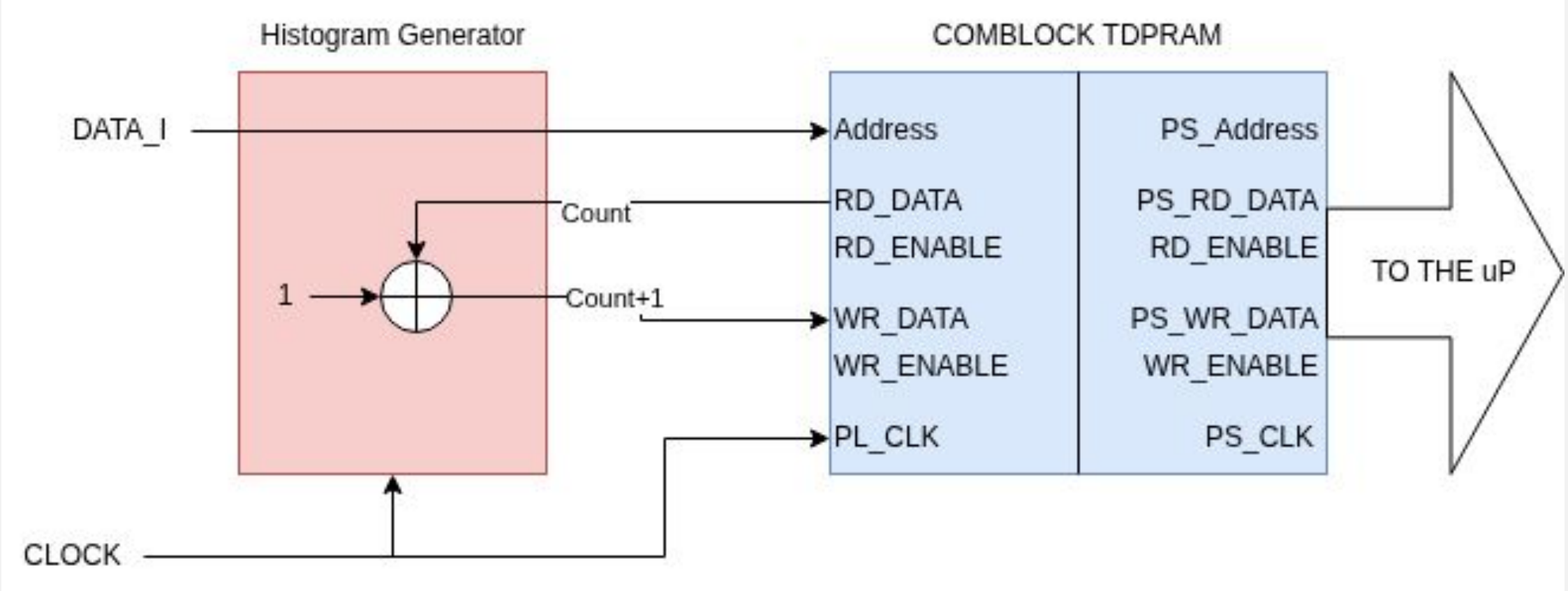
# Applying it in our design.



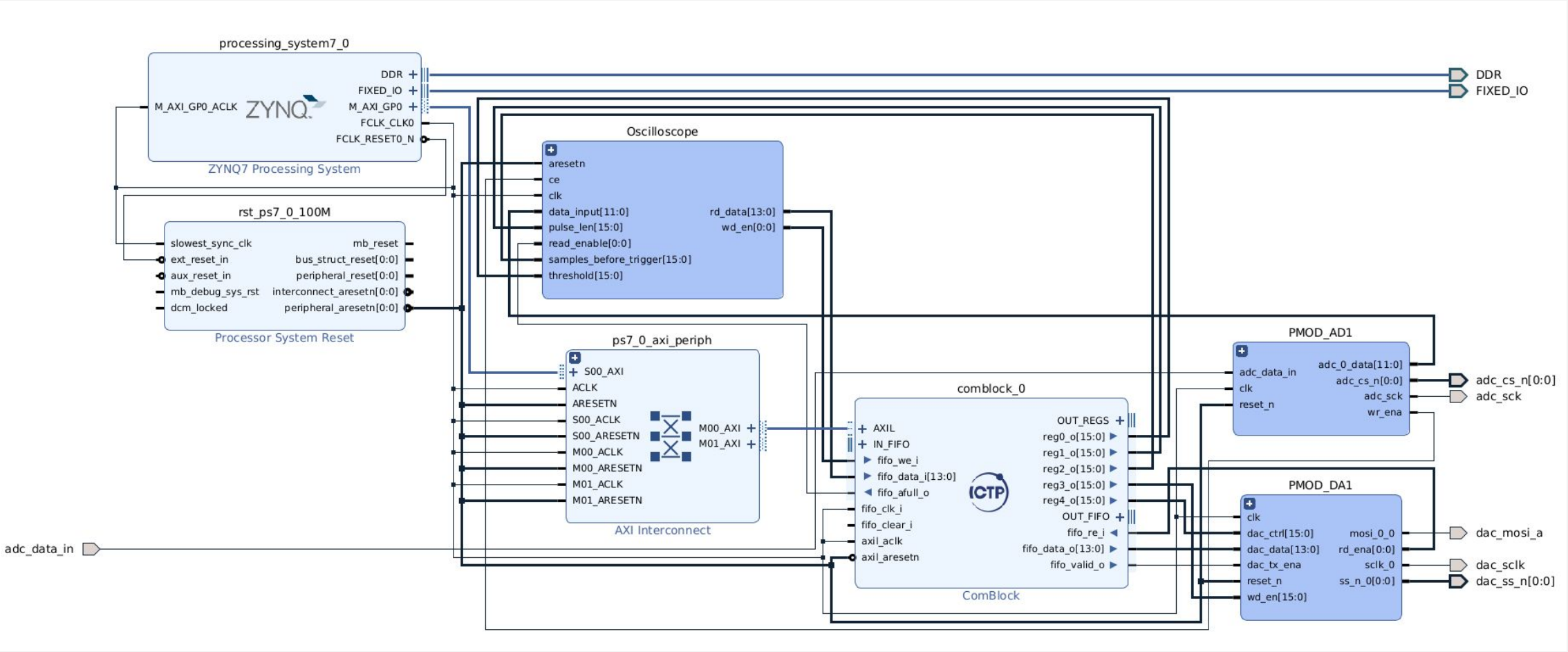
# Quick recap


oscilloscope
Data In
Cross Level Triger
One Shot Timer
Delay with shift register
Ring Buffer
Data Out

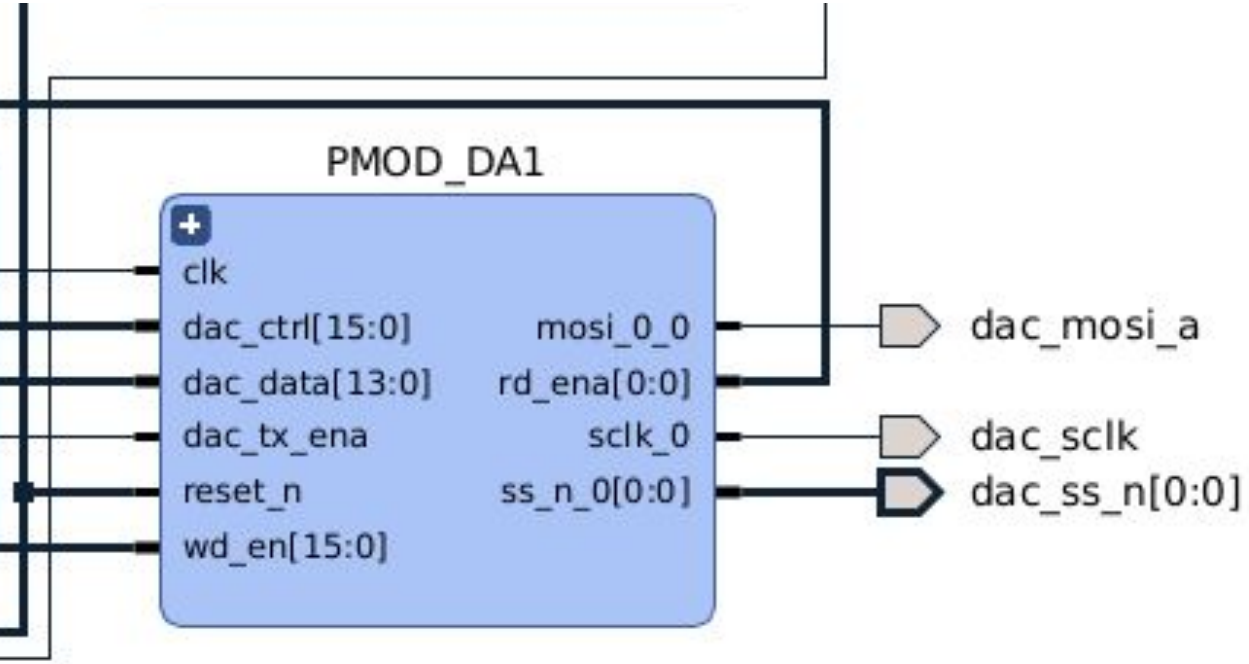
# How an histogram block works:



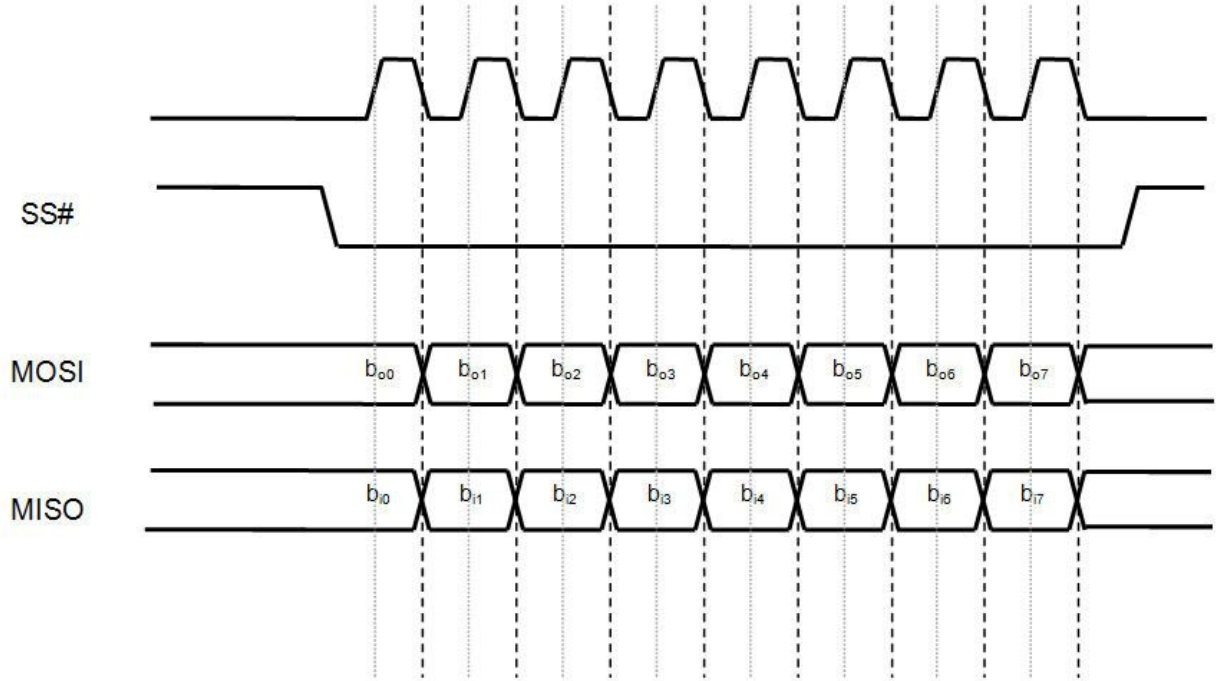
# Our Project


processing_system7_0
DDR
FIXED_IO
M_AXI_GP0_ACLK
M_AXI_GP0
FCLK_CLK0
FCLK_RESET0_N
ZYNQ7 Processing System
rst_ps7_0_100M
slowest_sync_clk
ext_reset_in
aux_reset_in
mb_debug_sys_rst
dcm_locked
mb_reset
bus_struct_reset[0:0]
peripheral_reset[0:0]
interconnect_aresetn[0:0]
peripheral_aresetn[0:0]
Processor System Reset
Oscilloscope
aresetn
ce
clk
data_input[11:0]
pulse_len[15:0]
read_enable[0:0]
samples_before_trigger[15:0]
threshold[15:0]
rd_data[13:0]
wd_en[0:0]
ps7_0_axi_periph
S00_AXI
ACLK
ARESETN
S00_ACLK
S00_ARESETN
M00_ACLK
M00_ARESETN
M01_ACLK
M01_ARESETN
M00_AXI
M01_AXI
AXI Interconnect
comblock_0
AXIL
IN_FIFO
fifo_we_i
fifo_data_i[13:0]
fifo_afull_o
fifo_clk_i
fifo_clear_i
axil_aclk
axil_aresetn
OUT_REGS
reg0_o[15:0]
reg1_o[15:0]
reg2_o[15:0]
reg3_o[15:0]
reg4_o[15:0]
OUT_FIFO
fifo_re_i
fifo_data_o[13:0]
fifo_valid_o
ICTP
ComBlock
PMOD_AD1
adc_data_in
clk
reset_n
adc_0_data[11:0]
adc_cs_n[0:0]
adc_sck
wr_ena
PMOD_DA1
clk
dac_ctrl[15:0]
dac_data[13:0]
dac_tx_ena
reset_n
wd_en[15:0]
mosi_0_0
rd_ena[0:0]
sclk_0
ss_n_0[0:0]
DDR
FIXED_IO
adc_cs_n[0:0]
adc_sck
adc_data_in
dac_mosi_a
dac_sclk
dac_ss_n[0:0]

# Serial Peripheral Interface (SPI)







# What is TCL?

TCL (Tool Command Language) or ***“Tickle”*** is a powerful, flexible scripting language initially developed in the late **1980**s by John Ousterhout. It's designed for command scripting and rapid prototyping.

- **Key Characteristics**:
  - **Interpreted Language**: TCL scripts are executed line-by-line, making it easy to test and modify in real-time.
  - **Cross-Platform Compatibility**: Runs on various operating systems, making it versatile across different development environments.
  - **Extensible**: TCL supports integration with other languages and tools, allowing it to adapt to various workflows.

# Why TCL is relevant?

**Widely Used in FPGA Tools**: Major FPGA design tools, such as Xilinx Vivado, Intel Quartus, and Microsemi Libero, include TCL as their primary scripting language.

**Automation and Control**: TCL scripts can control FPGA design tools for tasks like synthesis, simulation, and implementation. It allows engineers to automate complex workflows that would be tedious and error-prone if done manually.

**Adaptability**: TCL provides a way to customize and streamline the FPGA design process, making it easier to adapt the workflow to specific project requirements and increase productivity.

# TCL is both ingenious and frustrating

Tcl interpreters follow a basic set of rules, and that’s what makes it a good tool command language in the first place.

Everything is a string:

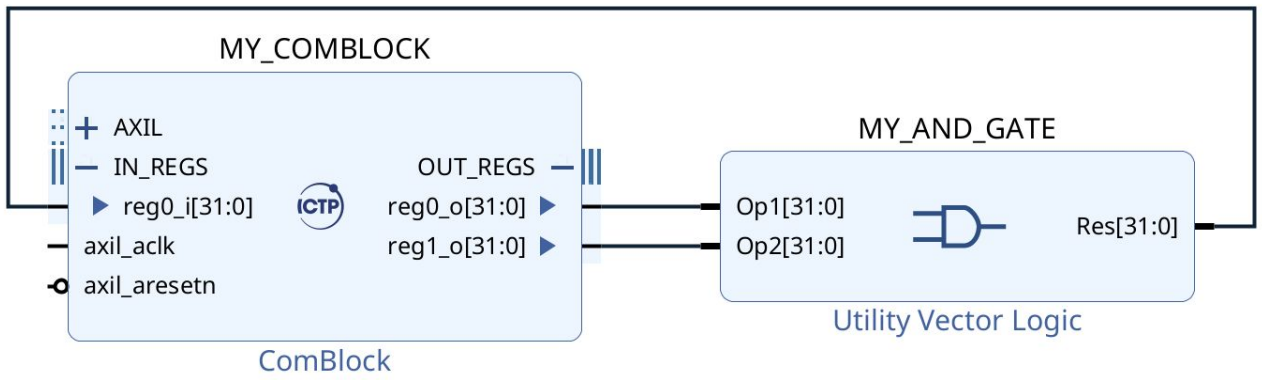
```
% set listA [list 1 2 3]
1 2 3
% set listB "1 2 3"
1 2 3
% string match $listA $listB
1
%
```

# Everything can be redefined

```
% puts "Hello World!"
Hello World!
%
% # Let's redefine puts to do something more
% rename puts puts_orig
% proc puts {args} {puts_orig "BMAZED! $args"}
%
% puts "Hello World!"
BMAZED! {Hello World!}
%
```

# Role of TCL in FPGA Design Flow

- **Automating Repetitive Tasks:** SoC design involves many repetitive tasks, such as defining constraints, running synthesis, or generating reports. TCL scripts can be written to perform these tasks automatically. E.g. Synthesis, implementation, and generate a bitstream.
- **Configuration and Setup:**TCL scripts are frequently used to configure the design environment, including initializing settings and setting up design constraints, paths, and other variables. E.g. Board selection, Constraints (timing, placement, etc).
- **Synthesis, Simulation, and Implementation**
- **Batch Processing:** Multiple simulations, implementations, or tests need to be run on different configurations or parameters.



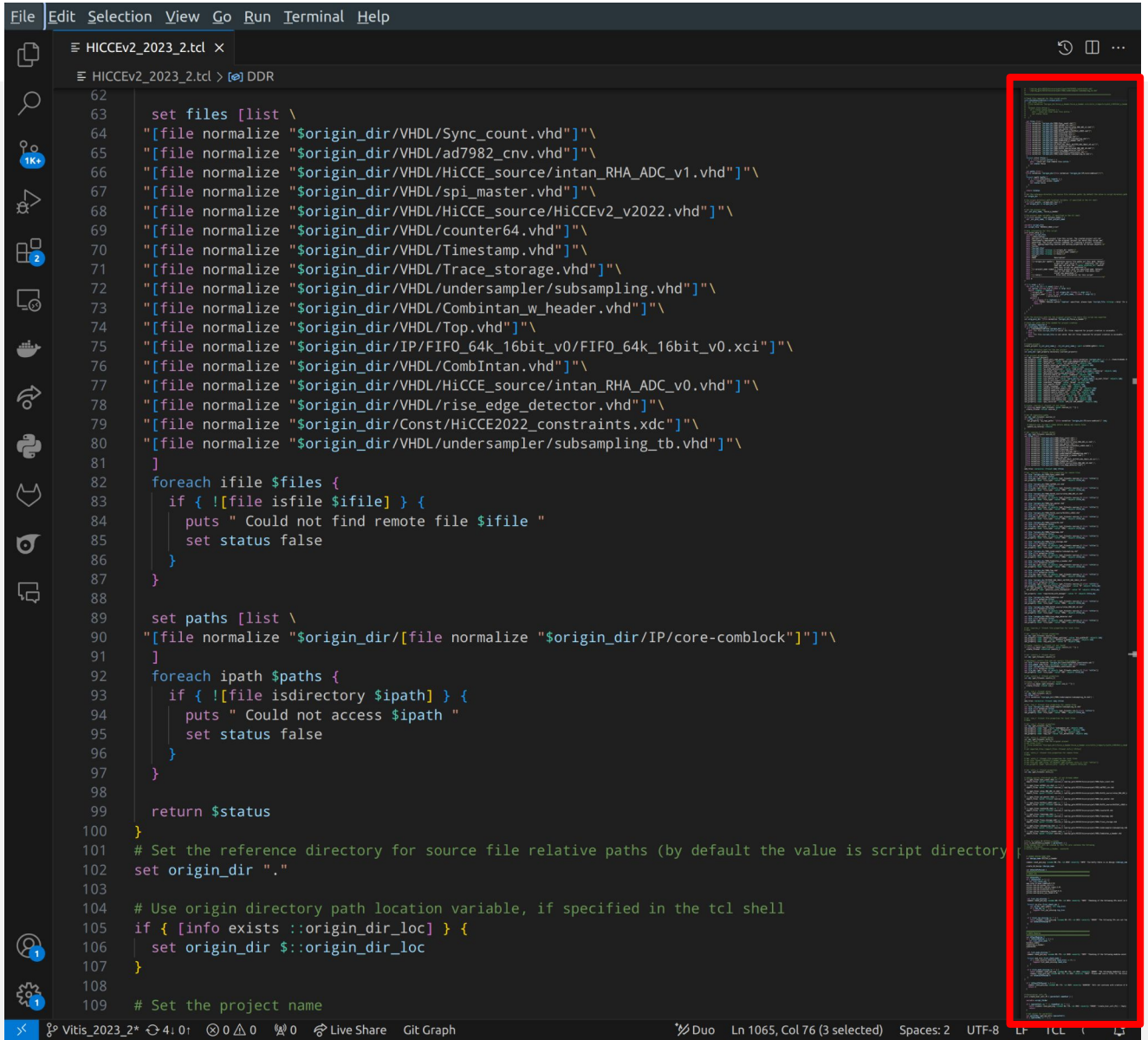

# TCL Example

```
### Instantiating Comblock
create_bd_cell -type ip -vlnv www.ictp.it:user:comblock:2.0 MY_COMBLOCK
set_property -dict [list \
 CONFIG.REGS_IN_DEPTH {1} \
 CONFIG.REGS_OUT_DEPTH {2} \
 CONFIG.DRAM_IO_ENA {false} \
 CONFIG.FIFO_IN_ENA {false} \
] [get_bd_cells MY_COMBLOCK]
### Instantiating Flopoco division module
create_bd_cell -type ip -vlnv xilinx.com:ip:util_vector_logic:2.0 MY_AND_GATE
set_property CONFIG.C_SIZE {32} [get_bd_cells MY_AND_GATE]

# Block Diagarm Interconnection
# Comblock to Logic Vector
connect_bd_net [get_bd_pins MY_COMBLOCK/reg0_o] [get_bd_pins MY_AND_GATE/Op1]
connect_bd_net [get_bd_pins MY_COMBLOCK/reg1_o] [get_bd_pins MY_AND_GATE/Op2]
# Logic Vector to Comblock
connect_bd_net [get_bd_pins MY_AND_GATE/Res] [get_bd_pins MY_COMBLOCK/reg0_i]
```

## The Frustrating Part

# It can get very complicated very quickly.

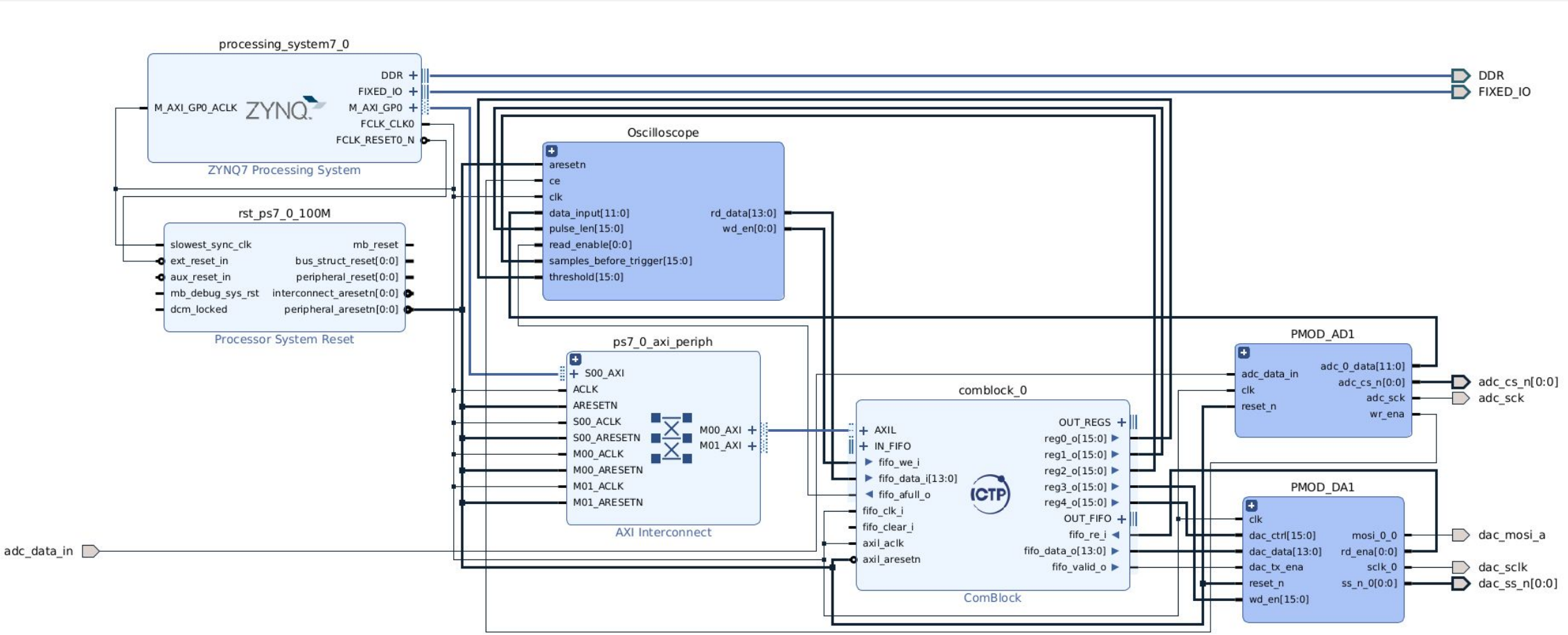


**You haven't answered the question….**

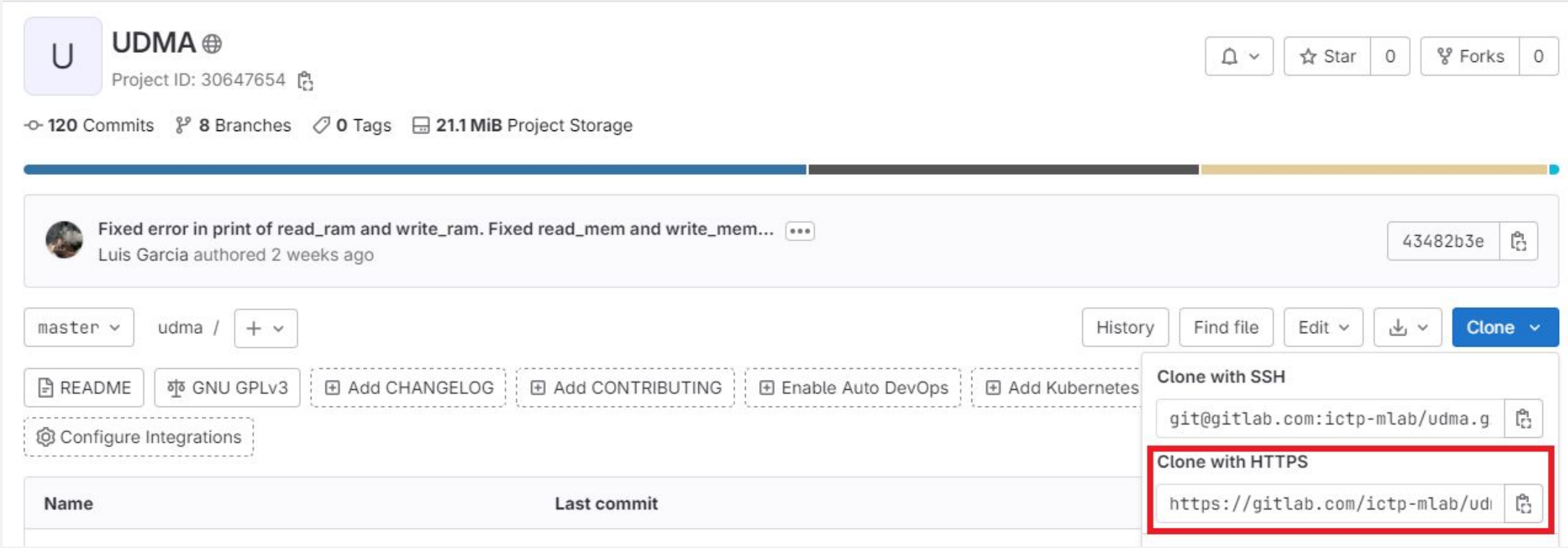
# Why TCL?

## Great for automation flow

```
source ./bd_comblock.tcl
```



# UDMA Library



# UDMA Library

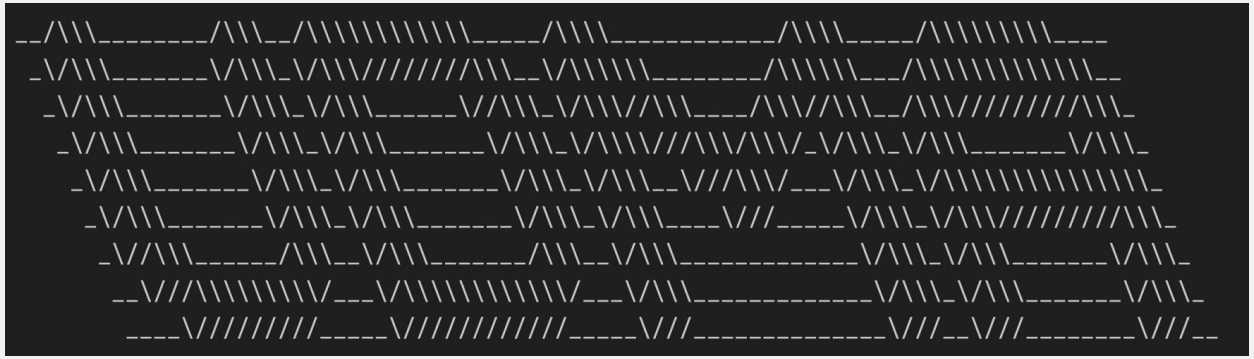
**Attributes**

**---**

**ip: str**

**port: int**

**s: socket**



**Methods**

**---**

**set_ip(ip)**

**set_port(port)**

**disconnect()**

**get_ip()**

**get_port()**

**connect()**

**close_server()**

**read_reg(reg)**

**write_reg(reg, data)**

**read_fifo(N)**

**write_fifo(N, data)**

**read_ram(addr, length, inc)**

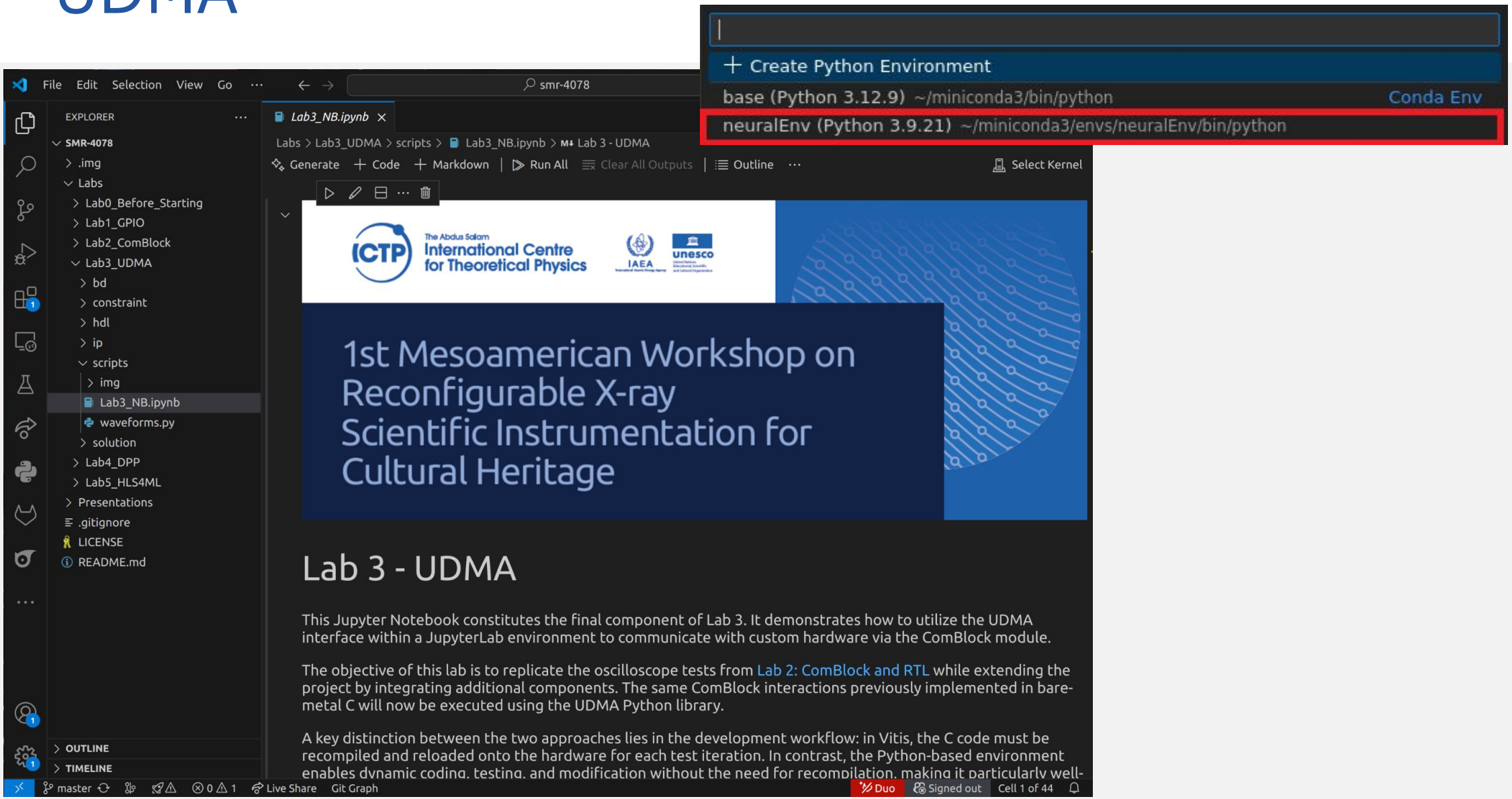
**write_ram( addr, length, inc, data)**

**read_mem(addr, length, inc)**

**write_mem(addr, length, inc, data)**

**select_comblock()**

# UDMA


Create Python Environment
base (Python 3.12.9) ~/miniconda3/bin/python
Conda Env
neuralEnv (Python 3.9.21) ~/miniconda3/envs/neuralEnv/bin/python
File Edit Selection View Go
smr-4078
EXPLORER
Lab3_NB.ipynb
Labs > Lab3_UDMA > scripts > Lab3_NB.ipynb > M↓ Lab 3 - UDMA
Generate Code Markdown Run All Clear All Outputs Outline
Select Kernel
SMR-4078
.img
Labs
Lab0_Before_Starting
Lab1_GPIO
Lab2_ComBlock
Lab3_UDMA
bd
constraint
hdl
ip
scripts
img
Lab3_NB.ipynb
waveforms.py
solution
Lab4_DPP
Lab5_HLS4ML
Presentations
.gitignore
LICENSE
README.md
OUTLINE
TIMELINE
The Abdus Salam International Centre for Theoretical Physics
IAEA
unesco
1st Mesoamerican Workshop on Reconfigurable X-ray Scientific Instrumentation for Cultural Heritage
Lab 3 - UDMA
This Jupyter Notebook constitutes the final component of Lab 3. It demonstrates how to utilize the UDMA interface within a JupyterLab environment to communicate with custom hardware via the ComBlock module.
The objective of this lab is to replicate the oscilloscope tests from Lab 2: ComBlock and RTL while extending the project by integrating additional components. The same ComBlock interactions previously implemented in bare-metal C will now be executed using the UDMA Python library.
A key distinction between the two approaches lies in the development workflow: in Vitis, the C code must be recompiled and reloaded onto the hardware for each test iteration. In contrast, the Python-based environment enables dynamic coding, testing, and modification without the need for recompilation, making it particularly well-
master
Live Share
Git Graph
Duo
Signed out
Cell 1 of 44

# We will continue in the lab.

## 1st Mesoamerican Workshop on Reconfigurable X-ray Scientific Instrumentation for Cultural Heritage

**Luis Guillermo García Ordóñez**

IAEA International Atomic Energy Agency
*Atoms for Peace and Development*

IUPAP Centenary YEARS 1922 - 2022

Gobierno de la República Guatemala
Secretaría Nacional de Ciencia y Tecnología

Academia Caroliana Orbis Conspicua Interfaces Guatemalensis

ICTP The Abdus Salam International Centre for Theoretical Physics

**You haven’t answered the question….**

# Why TCL?

## Great for automation flow