



Del Algoritmo al Hardware:  
Aprendizaje Automático en Sistemas Embebidos

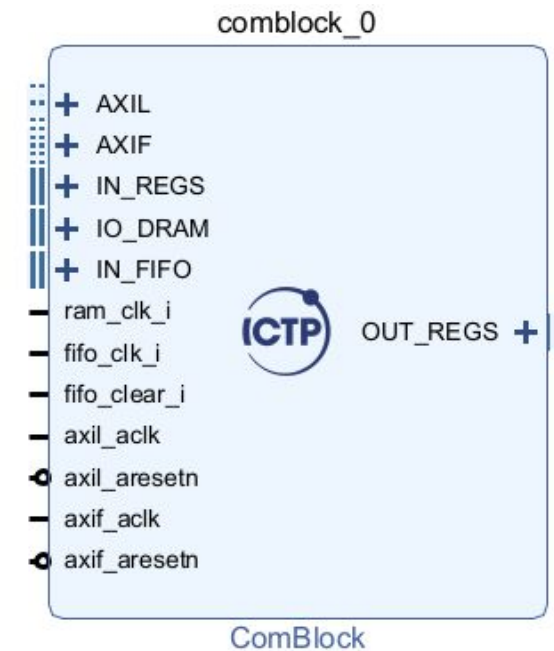
# *Core ComBlock*

Maynor Ballina, Abril 2025

# ¿Qué es el ComBlock?

El ComBlock es un bloque de comunicación que abstrae al usuario del protocolo AXI, simplificando así el diseño e integración de sistemas digitales.

- Interconexión con AXI: Compatible con interfaces comunes, como registros, memorias RAM y FIFOs.
- Recursos configurables: Permite ajustar parámetros según las necesidades del proyecto.
- Cruce de dominios de reloj: Incluye mecanismos integrados para el cruce de dominios de reloj utilizando FIFOs y memoria RAM de doble puerto.
- Driver: Se proporciona un driver en lenguaje C que facilita la integración completa del bloque.



MAYNOR BALLINA

Abril 2025



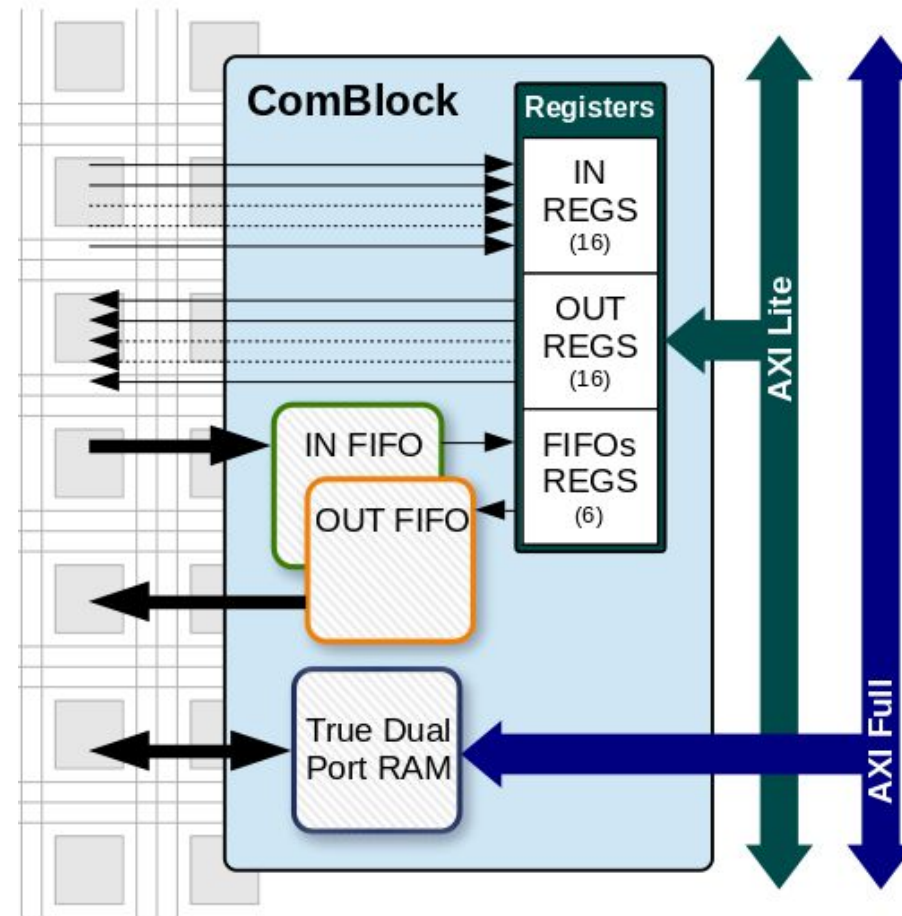
# Características del ComBlock

## Interfaces de Entrada/Salida:

- ❑ Registros de I/O
- ❑ FIFOs de I/O
- ❑ Memoria TDPRAM (True Dual Port RAM)

## Compatibilidad con AXI:

- ❑ AXI Lite: Soporta acceso a registros y FIFOs mediante esta interfaz ligera y de bajo ancho de banda.
- ❑ AXI Full: Utilizado para acceder a TDPRAM, ideal para transferencias de datos más grandes y de mayor rendimiento.



MAYNOR BALLINA

Abril 2025



# Registros I/O

- 16 registros de entrada: Comunicación de la FPGA hacia el procesador.
- 16 registros de salida: Comunicación del procesador hacia la FPGA.

## Configuración flexible:

- Ancho de datos configurable por el usuario: de 1 a 32 bits, según los requerimientos de la aplicación.
- Habilitación independiente de registros de entrada y salida: permitiendo un diseño más optimizado y adaptado a diferentes necesidades de comunicación.



Diagram showing the AXIL interface components:

- AXIL** (top level)
- IN\_REGS** (Input Registers):
  - reg0\_i[31:0]
  - reg1\_i[31:0]
  - axil\_ack
  - axil\_aresetn
- OUT\_REGS** (Output Registers):
  - reg0\_o[15:0]
  - reg1\_o[15:0]
  - reg2\_o[15:0]

### Registers

Input (FPGA to PROC)		Output (PROC to FPGA)	
<input checked="" type="checkbox"/> Enable		<input checked="" type="checkbox"/> Enable	
Data Width	32 [1 - 32]	Data Width	16 [1 - 32]
Quantity	2 [1 - 16]	Quantity	3 [1 - 16]

MAYNOR BALLINA

Abril 2025





# FIFOs

**FIFOs de Entrada y Salida:** Entrada (FPGA → Procesador), Salida (Procesador → FPGA)

**Ancho y profundidad personalizables:** Se puede configurar tanto el ancho de datos como la profundidad de las FIFOs.

**Funcionamiento asíncrono:** Las FIFOs permiten el cruce seguro entre dominios de reloj diferentes, facilitando la integración en sistemas heterogéneos.

**Registros de control del lado del procesador:** Lectura de nivel de ocupación, Reseteo o limpieza de las colas.

**FIFOs**

Input (FPGA to PROC)		Output (PROC to FPGA)	
<input checked="" type="checkbox"/> Enable		<input checked="" type="checkbox"/> Enable	
Data Width	16 [1 - 32]	Data Width	8 [1 - 32]
Depth	1024	Depth	256
Almost Full Offset	1	Almost Full Offset	1
Almost Empty Offset	1	Almost Empty Offset	1

MAYNOR BALLINA

Abril 2025



# TDPRAM (True Dual Port RAM)

**Acceso bidireccional simultáneo:** FPGA y procesador pueden realizar operaciones de lectura y escritura de forma independiente y concurrente.

**Cruce de dominios de reloj (Clock Domain Crossing):** Diseñada para funcionar en sistemas con distintos relojes, permitiendo una transferencia de datos segura y sin pérdida.

**Parámetros configurables por el usuario:** Ancho de datos, Ancho de dirección, Profundidad.

Ejemplo: ancho de dirección es de 10 bits, la profundidad por defecto será de  $2^{10} = 1024$ .

**True Dual Port RAM (FPGA to PROC, PROC to FPGA)**

- ☒ Enable
- Data Width: 8 [1 - 32]
- Address Width: 16 [1 - 32]
- Depth: 0

MAYNOR BALLINA

Abril 2025



# Bare Metal, C Driver

Información del dispositivo **xparameters.h**.

Register mappings en **combblock.h**.

Read and write functions **single memory position**:

```
void cbWrite(UINTPTR baseaddr, u32 reg, u32 value)
```

```
u32 cbRead(UINTPTR baseaddr, u32 reg)
```

Read and write several **contiguous memory positions**:

```
void cbWriteBulk(UINTPTR baseaddr, int *buffer, u32 depth)
```

```
void cbReadBulk(int *buffer, UINTPTR baseaddr, u32 depth)
```

MAYNOR BALLINA

Abril 2025



# UDMA

UDMA es una interfaz remota entre una PC y lógica personalizada en un SoC-FPGA, probada sobre FreeRTOS y LwIP, Libreria de Python - TCP socket wrapper.

set\_ip(ip)

set\_port(port)

connect()

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)

MAYNOR BALLINA

Abril 2025





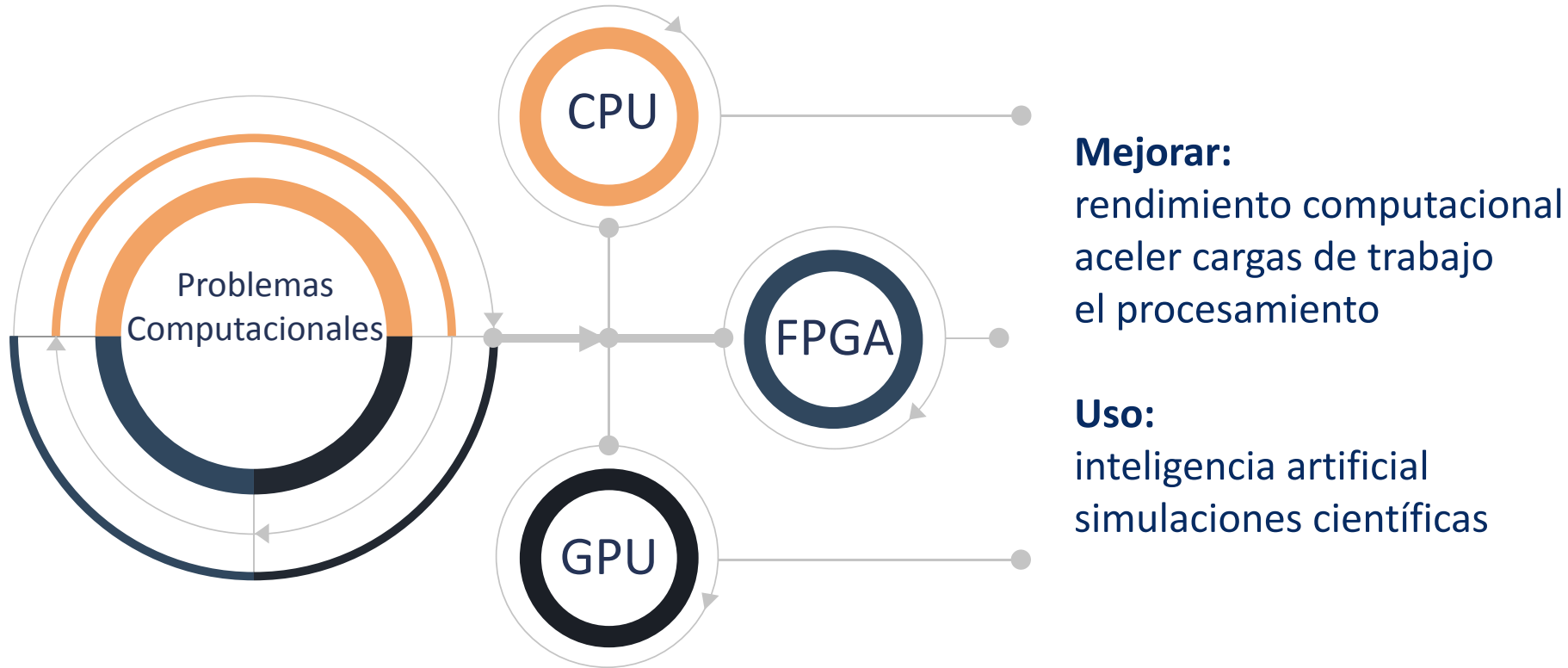


Del Algoritmo al Hardware:  
Aprendizaje Automático en Sistemas Embebidos

# *HyperFPGA: Computación heterogénea y distribuida en MPSoC-FPGA Cluster*

Maynor Ballina, Abril 2025

# ¿Qué es computación Heterogénea?



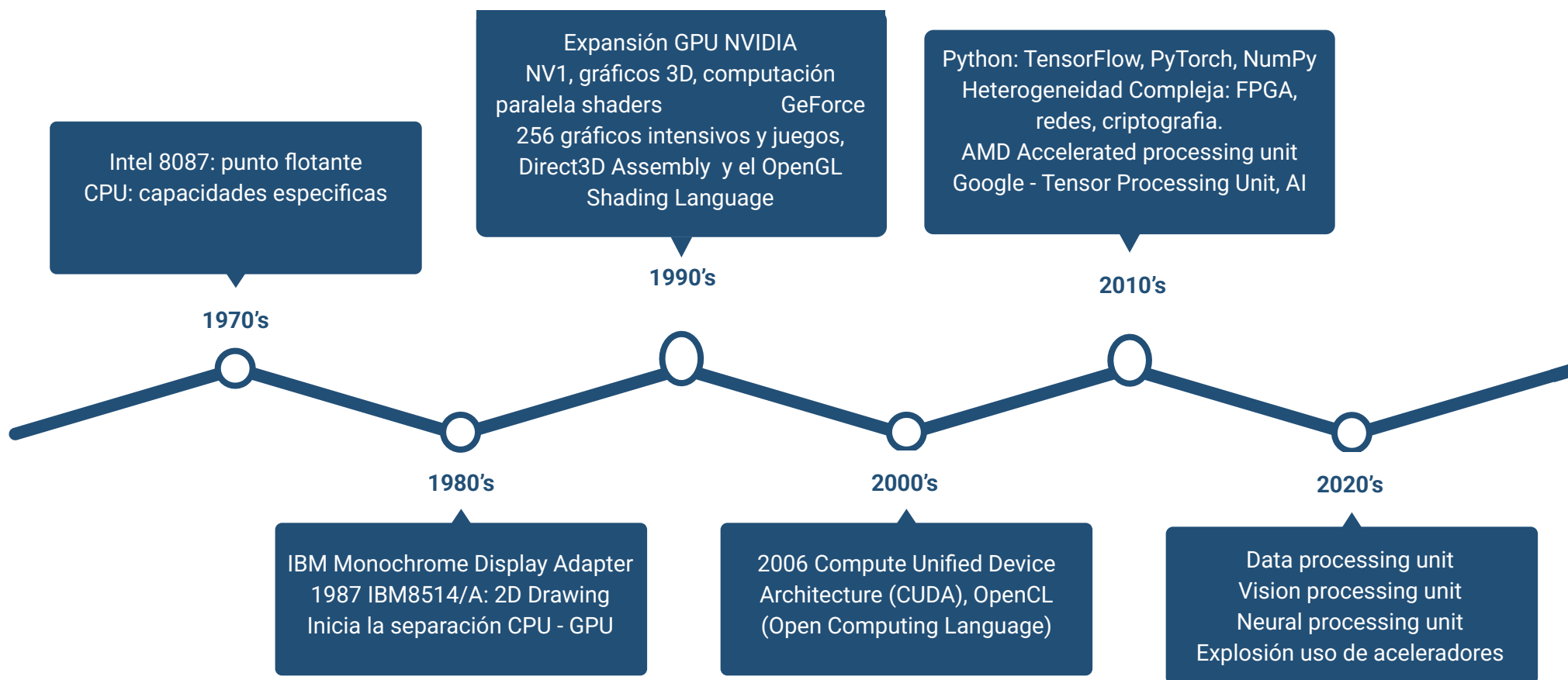
El uso coordinado de diferentes tipos de núcleos de procesamiento dentro de un mismo sistema para maximizar el rendimiento y la eficiencia energética.

MAYNOR BALLINA

Abril 2025



# Evolución de la computación heterogénea



MAYNOR BALLINA

Abril 2025



UNIVERSITÀ  
DEGLI STUDI  
DI TRIESTE

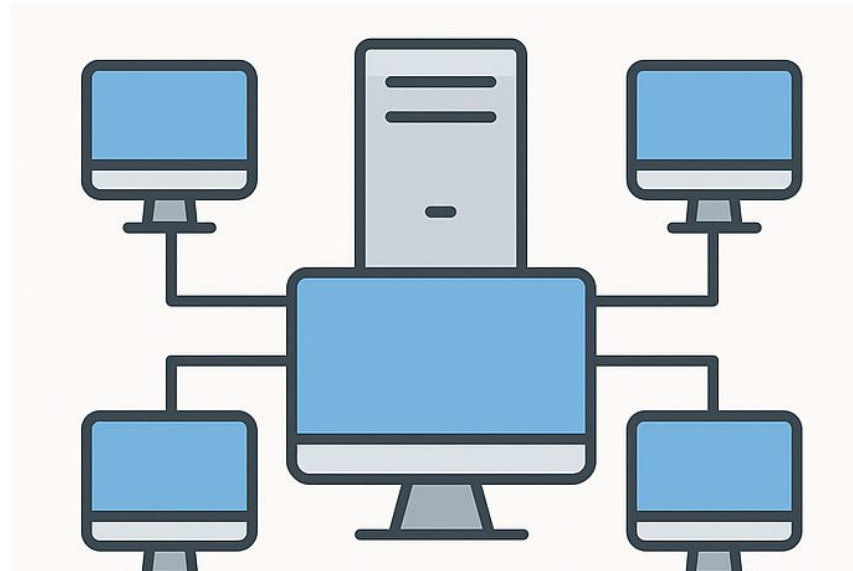


The Abdus Salam  
International Centre  
for Theoretical Physics

# Computación Distribuida

Un modelo en el que múltiples nodos trabajan juntos de manera coordinada para resolver un problema, compartiendo recursos para mejorar el rendimiento, la escalabilidad y la tolerancia a fallos.

La computación distribuida en clusters es un enfoque de procesamiento de datos que utiliza un conjunto de nodos, interconectados entre sí, para trabajar de manera conjunta como si fueran una sola máquina.



MAYNOR BALLINA

Abril 2025



# La taxonomía de Flynn

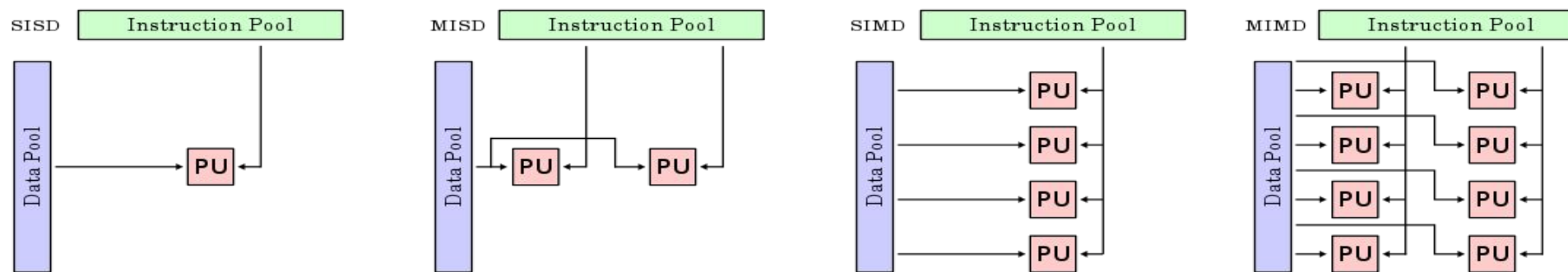
Clasifica las arquitecturas de computadoras según el número de flujos de instrucciones y flujos de datos.

SISD: Instrucción única, dato único - CPU convencional

SIMD: Instrucción única, múltiples datos - GPUS, extensiones de instrucciones en CPU

MISD: Múltiples instrucciones, único dato - Procesamiento en paralelo, detección errores

MIMD: Múltiples instrucciones, múltiples datos - Multiprocesador, supercomputadoras



Mike Flynn "Very High-Speed Computing Systems", Proc. Of IEEE, 1966

MAYNOR BALLINA

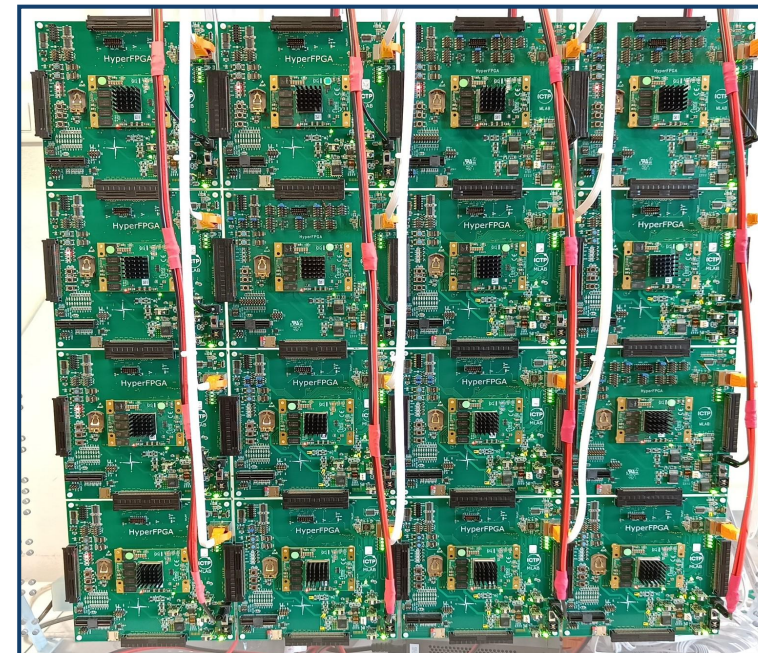
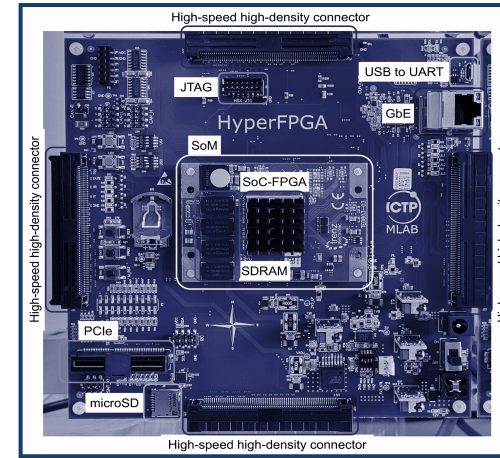
Abril 2025





# El HyperFPGA Cluster

- HyperFPGA Cluster
  - 16 Nodos de MPSoC SOM with AMD Zynq UltraScale+
  - Network: Ethernet connection TCP/IP, HP, HD GTH.
  - Debian Linux OS
- System-on-Module (Zynq UltraScale+ MPSoC-FPGA)
  - CPU: Quad Arm Cortex, Dual Arm Cortex
  - GPU: ARM Mali 400 MP2
  - FPGA: ZU4EG
- Carrier board
  - x4 PCIe expansion port, Jtag, USB-UART
  - microSD card
  - 4 High-speed connectors for FPGA direct Interconnection



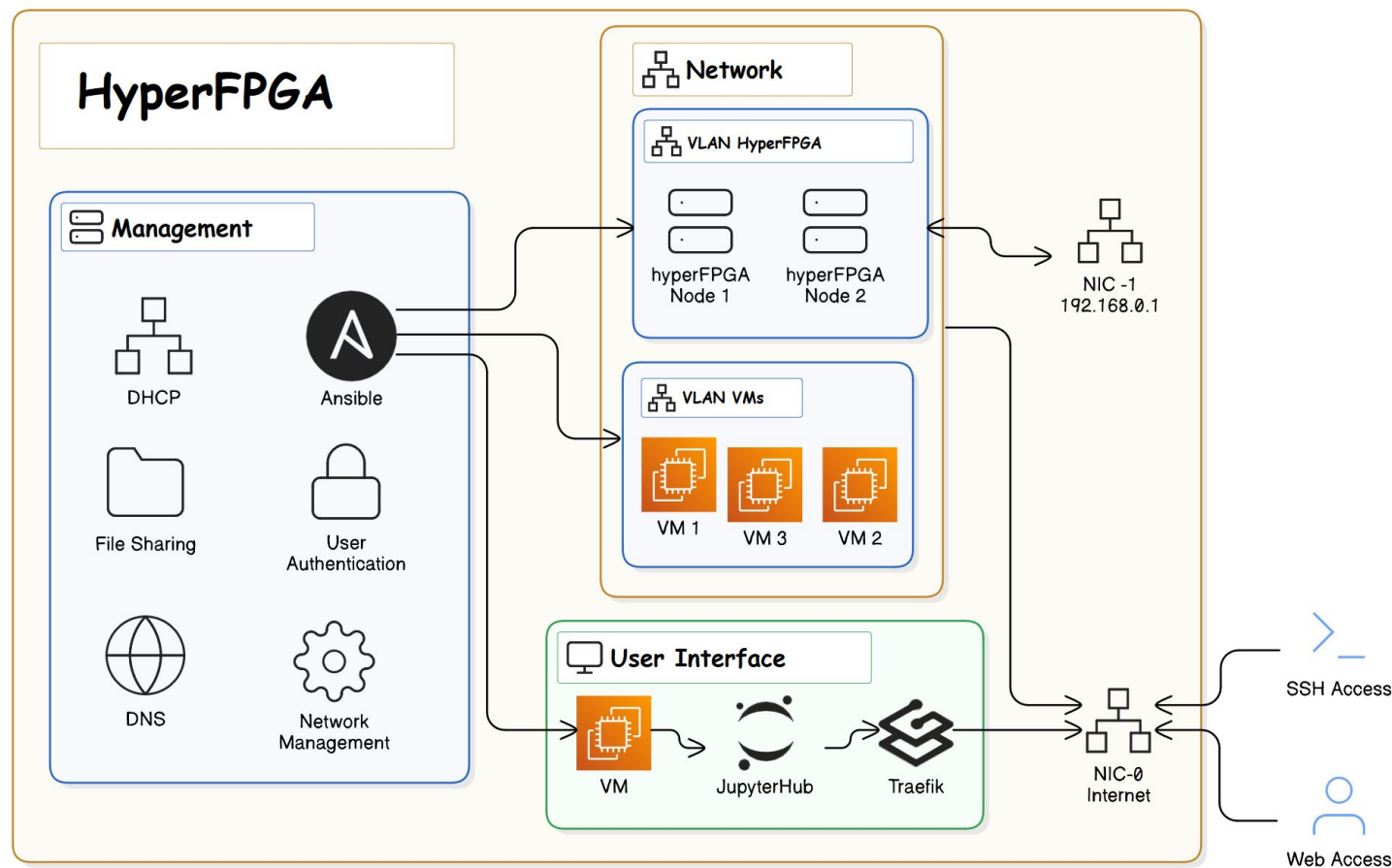
MAYNOR BALLINA

Abril 2025



# Cluster Management

- Infrastructure as Code (IaC).
- JupyterHub & Microservices for multi-user access.

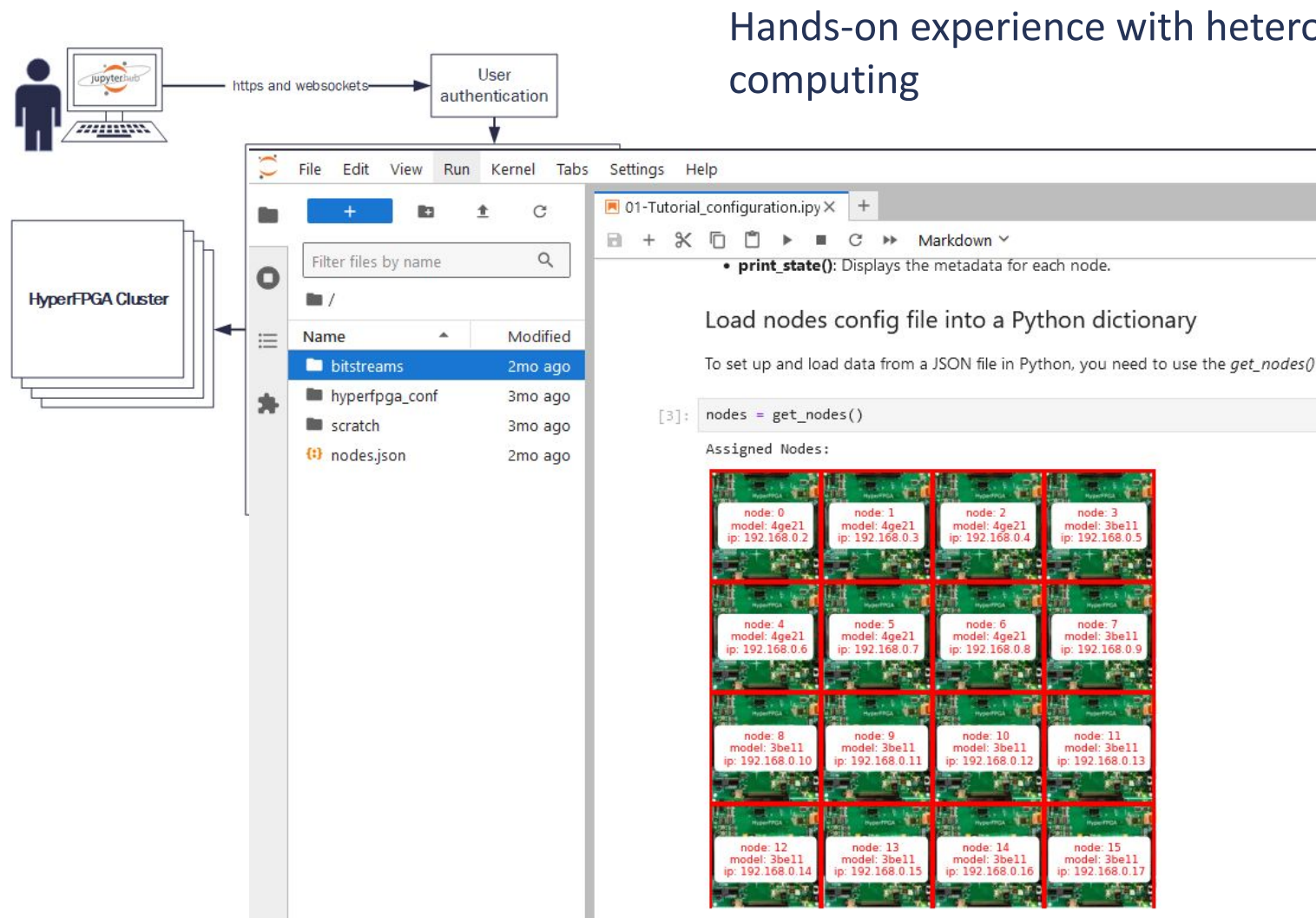


MAYNOR BALLINA

Abril 2025



# Remote Access to the Cluster



MAYNOR BALLINA

Abril 2025



The Abdus Salam  
International Centre  
for Theoretical Physics

# Remote Access to the Cluster

<https://hyperfpga.sti.ictp.it>

Sign In

Username:

Password:

Sign In

The screenshot shows the JupyterLab interface. On the left, the file browser displays a list of files and folders. A blue arrow points from the 'Getting\_Started' folder in the file browser to the 'Getting\_Started' folder in the launcher tab. The launcher tab shows a 'Notebook' section with two Python 3 (ipykernel) kernels, a 'Console' section with two Python 3 (ipykernel) kernels, and an 'Other' section with icons for Terminal, Text File, Markdown File, Python File, and Show Contextual Help.

Name	Modified
Getting_Started	2d ago
Welcome.pdf	12d ago

Name	Modified
bitstreams	21d ago
images	2d ago
01-Tutorial_conf...	12d ago
02-Tutorial_clust...	12d ago
basic_test-3be11...	21d ago
basic_test-4ge21...	21d ago

MAYNOR BALLINA

Abril 2025





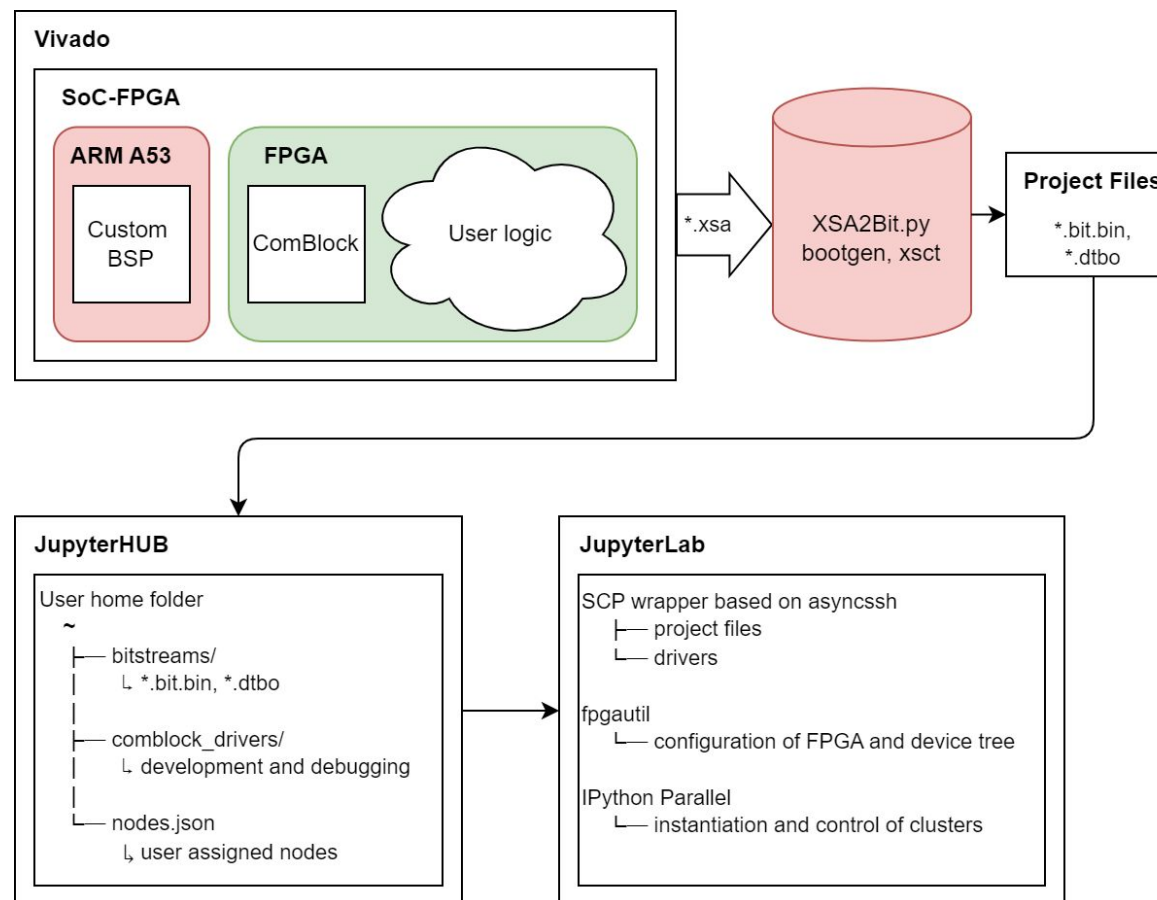
# Development Workflow

**Hardware definition:** Vivado  
Board support package  
ComBlock as abstraction layer  
between CPU and user logic

**Middleware:** XSA2Bit  
Generate device tree overlay from  
an XSA and compile sources

**Management:** JupyterHub  
Authentication  
Organized storage

**Programming:** JupyterLab  
User programmable context



MAYNOR BALLINA

Abril 2025





# Development Workflow - Vivado

## Default Part

Choose a default AMD part or board for your project.



Parts | **Boards**

**To fetch the latest available boards from git repository, click on 'Refresh' button.** [Dismiss](#)

[Reset All Filters](#)

Vendor:  Name:  Board Rev:

Search:

Display Name	Preview	Status	Vendor	File Version	Part	I/O Pin Count	Board
<a href="#">HyperFPGA 3be11</a>		Installed	ictp.it	1.0	xczu3eg-sfvc784-1-e	784	1.0
<a href="#">HyperFPGA 4ge21</a>		Installed	ictp.it	1.0	xczu4eg-sfvc784-2-e	784	1.0

**Export Hardware Platform**

**Files**

Enter the name of your hardware platform file, and the directory where the XSA file will be stored.

**-4ge21**

XSA file name:

**Export Hardware Platform**

**Files**

Enter the name of your hardware platform file, and the directory where the XSA file will be stored.

**-3be11**

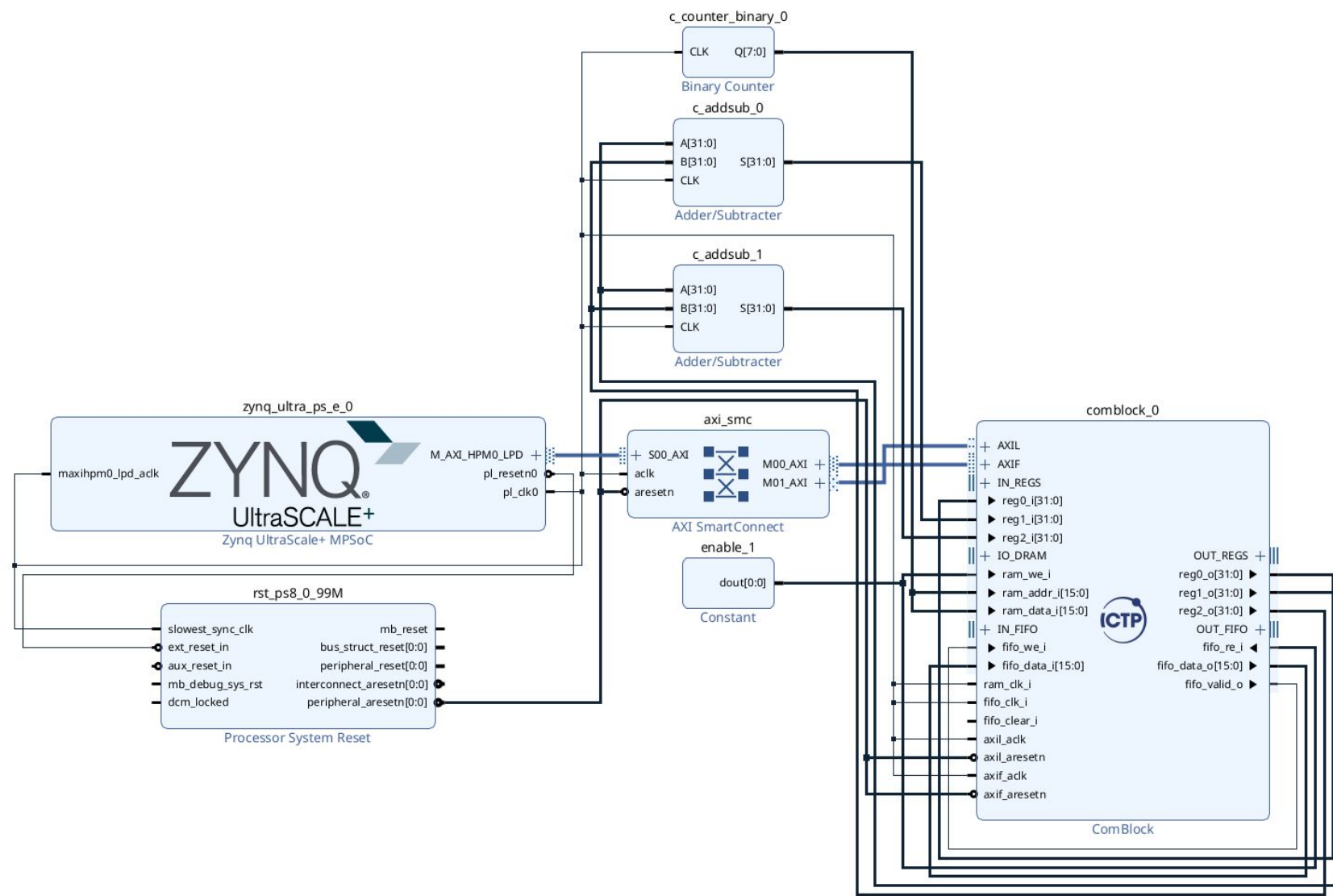
XSA file name:

MAYNOR BALLINA

Abril 2025



# Development Workflow - Vivado

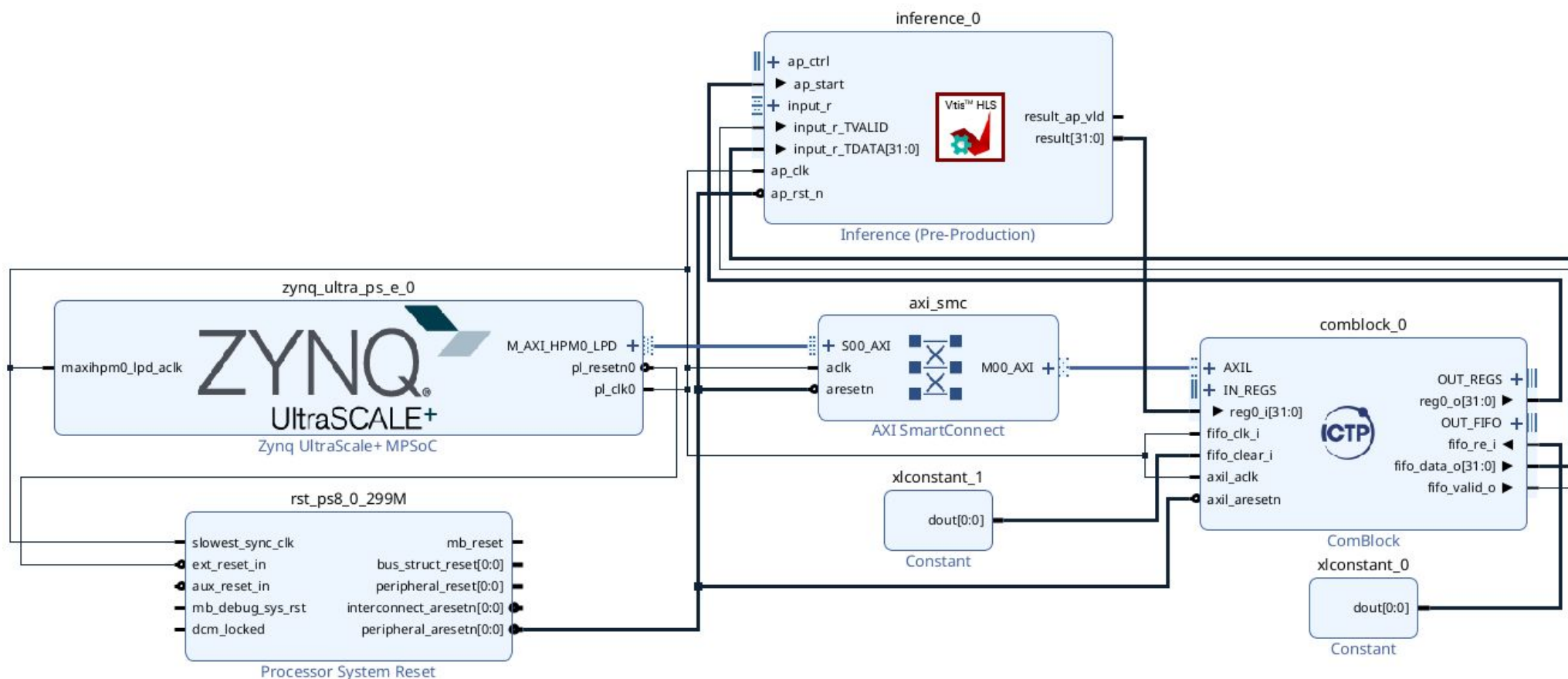


MAYNOR BALLINA

Abril 2025



# Development Workflow - Vivado

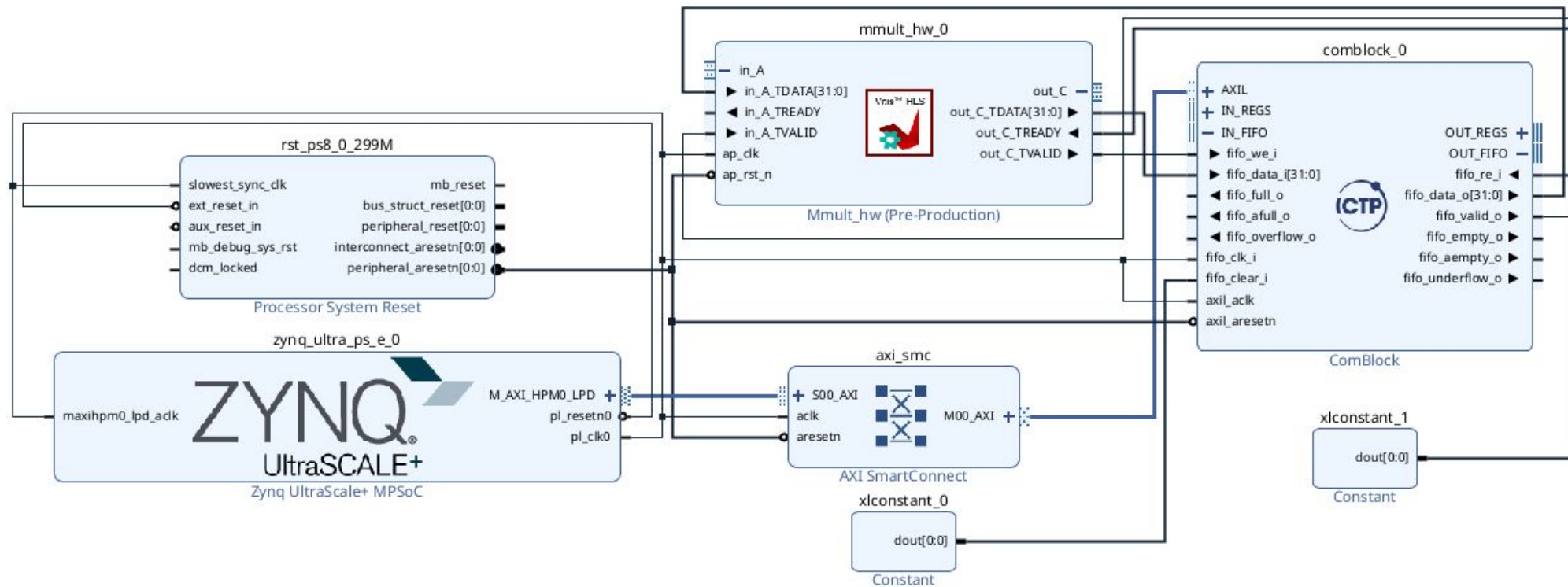


MAYNOR BALLINA

Abril 2025

UNIVERSITÀ  
DEGLI STUDI  
DI TRIESTEThe Abdus Salam  
International Centre  
for Theoretical Physics

# Development Workflow - Vivado



MAYNOR BALLINA

Abril 2025

# Development Workflow - Python

Central server (Local): `import hyperfpga_cluster as hfc`

Node (Remoto): `from comblock import Comblock - cb = Comblock()`

- IPython Parallel es una extensión de IPython que permite ejecutar código en paralelo en múltiples núcleos o incluso en varios nodos de una red.
- Objetivo: Mejorar el rendimiento de las tareas que requieren gran capacidad computacional mediante la paralelización.
- Características principales:
  - Ejecución concurrente de código en múltiples procesos.
  - Soporte para la distribución de tareas en diferentes máquinas (clusters).
  - Control centralizado de tareas y monitoreo de procesos.

MAYNOR BALLINA

Abril 2025





# Development Workflow - Python

## Controlador (Controller):

- Administra los recursos y distribuye las tareas a los "engines".
- Centraliza la comunicación entre los usuarios y los engines.

## Engines:

- Procesos que ejecutan las tareas distribuidas.
- Pueden correr en una máquina local o remota.

## Cliente:

- Herramienta que permite a los usuarios interactuar con el controlador y enviar tareas.
- Se puede usar desde un notebook de Jupyter.

## Flujo de trabajo:

- El cliente envía comandos al controlador.
- El controlador distribuye los comandos a los engines.
- Los engines ejecutan el código en paralelo y devuelven los resultados.

MAYNOR BALLINA

Abril 2025



# Development Workflow - Python

## Direct View (DirectView)

Un Direct View en ipyparallel es una forma de ejecutar comandos en todos los trabajadores al mismo tiempo.

`remote_client[:]` o `remote_client.direct_view()`, se crea un objeto `DirectView` que representa todos los nodos del cluster.

Cualquier código que ejecutes con este objeto se enviará simultáneamente a todos los trabajadores.

## Load-Balanced View (LoadBalancedView)

Un Load-Balanced View distribuye automáticamente las tareas entre los trabajadores, asegurándose de que la carga de trabajo esté equilibrada.

`rc.load_balanced_view()`, se crea un objeto `LoadBalancedView`. En lugar de enviar el mismo código a todos los trabajadores, `LoadBalancedView` distribuye tareas de manera eficiente.

MAYNOR BALLINA

Abril 2025



# Conclusión: ¿Por qué es necesaria la computación heterogénea?

- **Optimización del rendimiento:** Las GPU son superiores en procesamiento paralelo masivo, mientras que las CPU son mejores en tareas secuenciales complejas y de control. Las FPGA permiten una personalización específica de tareas críticas y optimizaciones en tiempo real.
- **Eficiencia energética:** Algunos aceleradores, como las FPGA, son más eficientes energéticamente que las CPU o GPU, lo que es crucial en simulaciones que requieren gran cantidad de recursos computacionales durante largos períodos de tiempo.
- **Escalabilidad:** Los aceleradores permiten que las simulaciones se ejecuten de manera eficiente en escalas más grandes. Sin computación heterogénea, algunas simulaciones que requieren el procesamiento de datos en tiempo real, o en escalas masivas, serían inviables.
- **Reducción de latencia:** En ciertos problemas, como los que requieren simulaciones en tiempo real o feedback inmediato, las FPGA y otros aceleradores pueden reducir significativamente la latencia del sistema.

MAYNOR BALLINA

Abril 2025



Maynor Ballina, Abril 2025



The Abdus Salam  
International Centre  
for Theoretical Physics



UNIVERSITÀ  
DEGLI STUDI  
DI TRIESTE

Muchas Gracias

Q&A

[maynorgiovanni.ballinaescobar@phd.units.it](mailto:maynorgiovanni.ballinaescobar@phd.units.it)

[mballina@ictp.it](mailto:mballina@ictp.it)

