

Présentation Projet R&D

Cosimulation HW/SW Multi-RISC V

10/04/2024



**ADVANS
GROUP**



ELSYS
DESIGN



AVISTO



MECAGINE

A decorative background at the top of the slide featuring a network of interconnected nodes and lines in shades of blue and white.

► Plan

- **Objectifs**
- **Environnement de test**
- **Application au projet multi RISC-V**
- **Application au test d'IPs et de drivers sur NG-Ultra**
- **Prochaines étapes**

► Objectif projet R&D

Tester et débbuger les fonctionnalités d'un système comprenant du HW et du SW rapidement afin d'accélérer le développement de/sur SoC.

Caractéristiques:

- **Bien plus rapide qu'une simulation HDL classique**
- **Offre bien plus de données qu'un test sur carte (waveforms)**
- **Possibilité de contrôler chaque cœur avec une instance de gdb**
- **Basé sur des logiciels et des bibliothèques open source**

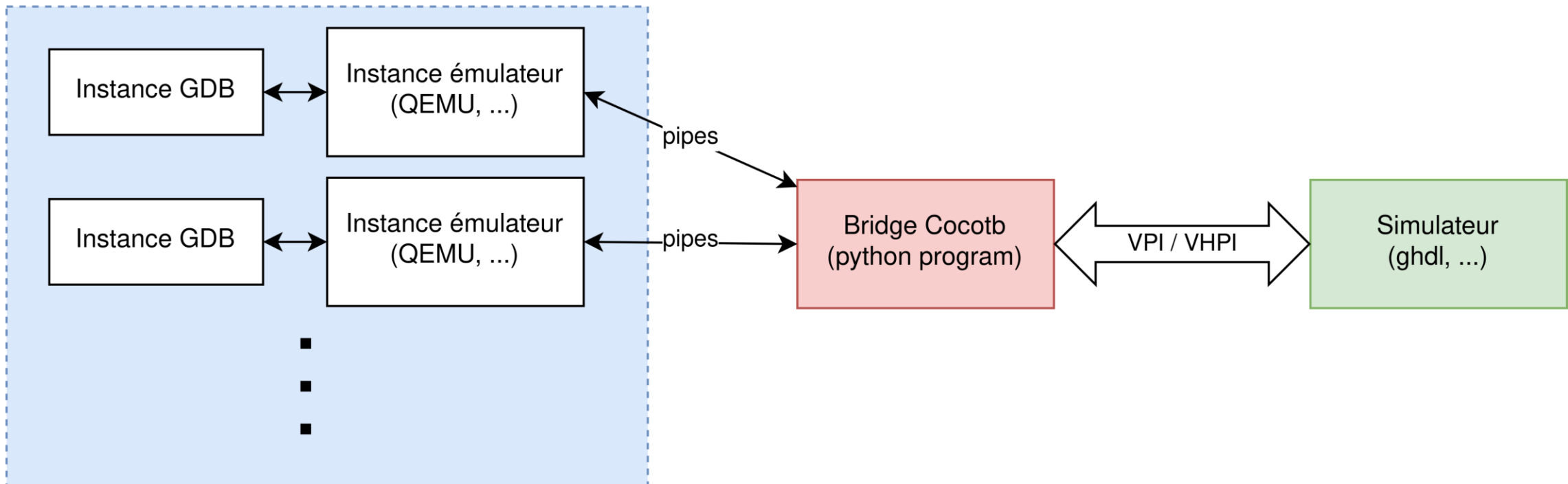
Limitations:

- **Cosimulation fonctionnelle mais pas exacte dans le temps**
- **Implémentation du support d'une nouvelle architecture relativement complexe**

A stylized profile of a human head facing right, filled with a blue-to-white gradient. Inside the head, a complex network of white dots connected by thin white lines is visible. The background is a light blue gradient with a vertical white stripe.

Environnement de test

► Environnement de test



▶ Généricité de l'environnement

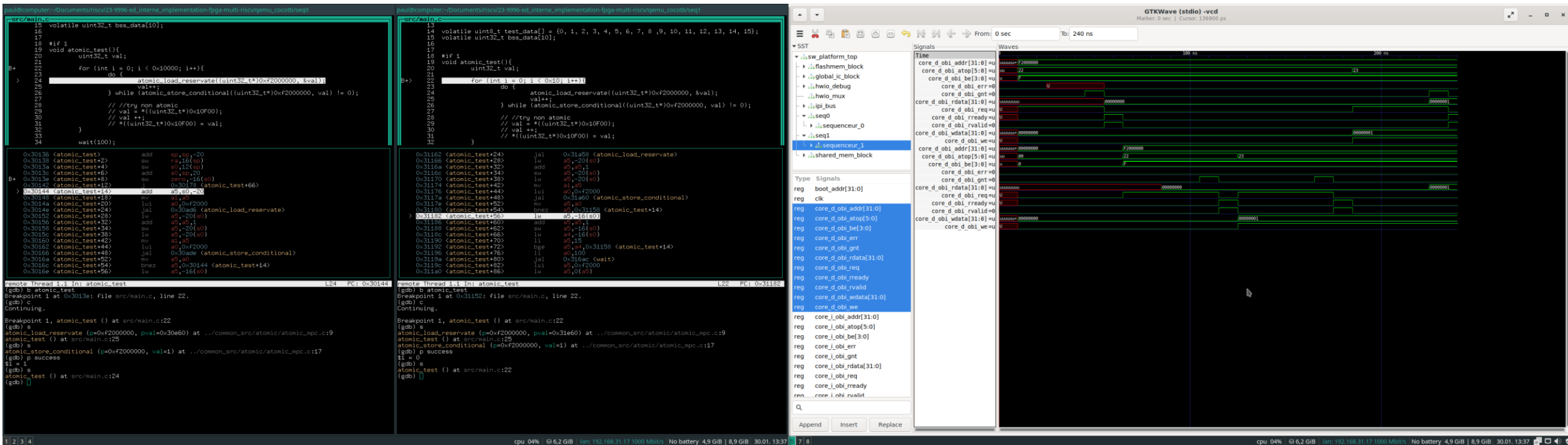
- L'interface QEMU <-> bridge Cocotb est générique: d'autres émulateurs pourraient être utilisés
 - Elle utilise plusieurs "named pipes" (POSIX) et suit un protocole basique d'accès mémoires et de levée d'interruptions, facilement portable.
- Cocotb permet d'utiliser de nombreux simulateurs sans changer le code du bridge

▶ Emulateur (QEMU):

- **Avantages de QEMU:**
 - Open source
 - En développement continu depuis plus de 15 ans
 - Emulateur connu
- **Modifications de QEMU:**
 - Ajout d'un "device" nommé hdl_bridge qui communique avec le bridge Cocotb à l'aide des pipes
 - Ajout d'une "machine" personnalisée pour chaque architecture supportée

Débogueur (GDB):

- Chaque cœur émulé peut être contrôlé par une instance GDB
- Les signaux peuvent être suivis (avec ghdl + gtkwave)



The image displays a development environment with three main components:

- Left Panel (GDB Console):** Shows the GDB interface with source code for `atomic_test.c` and the execution log. The code includes a loop for testing atomic operations on a shared memory block. The log shows the program running and hitting a breakpoint at line 22.
- Middle Panel (GDB Console):** Shows the GDB interface with source code for `atomic_test.c` and the execution log. The code includes a loop for testing atomic operations on a shared memory block. The log shows the program running and hitting a breakpoint at line 22.
- Right Panel (GTKWave):** Shows the signal tracing interface. It displays a list of signals on the left, including `core_d_obi_addr[31:0]`, `core_d_obi_atop[5:0]`, `core_d_obi_be[3:0]`, `core_d_obi_err`, `core_d_obi_gnt`, `core_d_obi_rdata[31:0]`, `core_d_obi_req`, `core_d_obi_ready`, `core_d_obi_rvalid`, `core_d_obi_wdata[31:0]`, and `core_d_obi_we`. The main area shows a timing diagram with multiple signals plotted against time, with a scale of 240 ns.

- **Avantages de cocotb:**
 - Open source
 - Offre une interface python générique avec le simulateur HDL (l'interface est la même pour tous les simulateurs)
 - Supporte de nombreux simulateurs
- **Le bridge implémente:**
 - La transformation des load / store (atomiques ou non) de QEMU en requêtes sur un bus (OBI, AXI, ...) dans la simulation VHDL
 - Le transfert de réponses aux load / store reçues de la simulation VHDL à QEMU
 - Le transfert de requêtes d'interruptions reçues de la simulation VHDL à QEMU

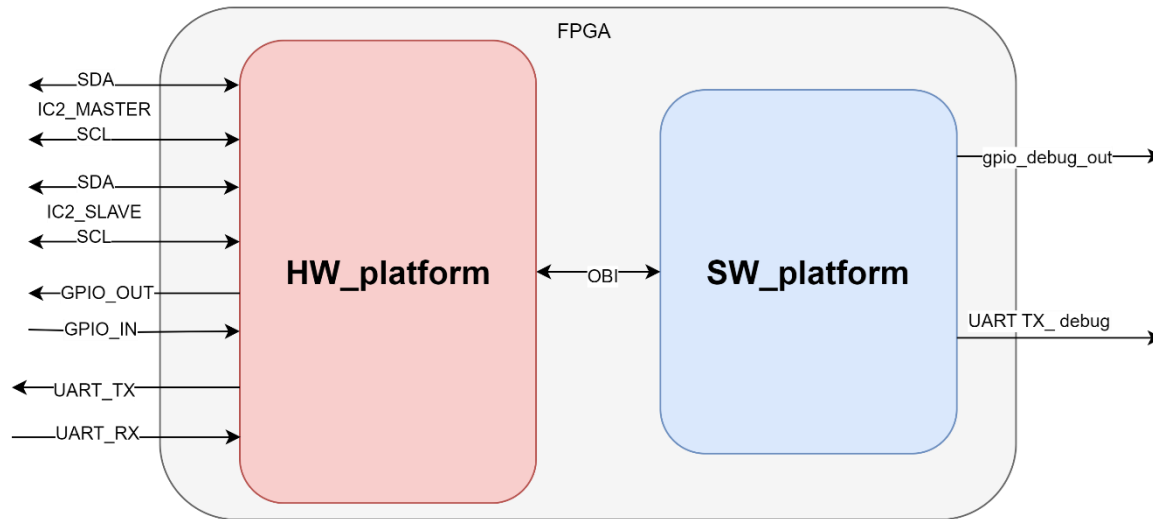
▶ Simulateur (GHDL):

- **Avantages de GHDL:**
 - Open source
 - Visualisation des signaux en direct avec GTKWave
- **Inconvénients:**
 - L'interface VPI de GHDL n'est pas complètement standard et cocotb ne supporte donc pas de lire des signaux de certains types. Il faut parfois créer un (mini) wrapper VHDL pour décomposer ces types en signaux simples.
 - Buffers pour les signaux VCD et GHW (besoin de patcher et recompiler GHDL pour avoir une visualisation des signaux directe).

A stylized profile of a human head facing right, filled with a blue-to-white gradient. Inside the head, a complex network of white dots connected by thin white lines represents a neural or data network. The background is a light blue gradient.

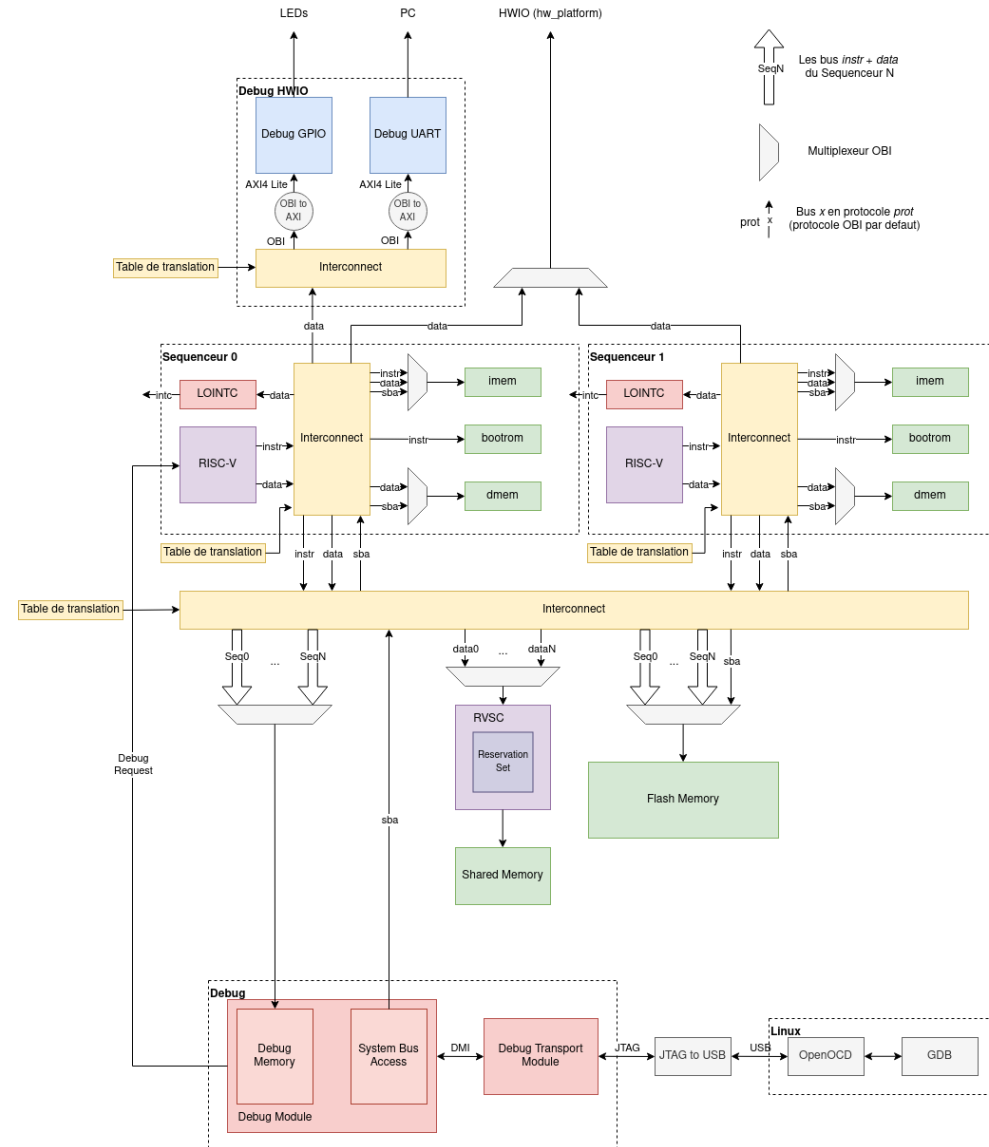
Application au projet multi RISCv

Architecture du design testé



- ▶ **HW Platform** : contient les modules IO et processing
- ▶ **SW Platform** : contient la partie processeurs RISC-V, c'est sur celle-là que l'on va se concentrer

Architecture de la SW platform

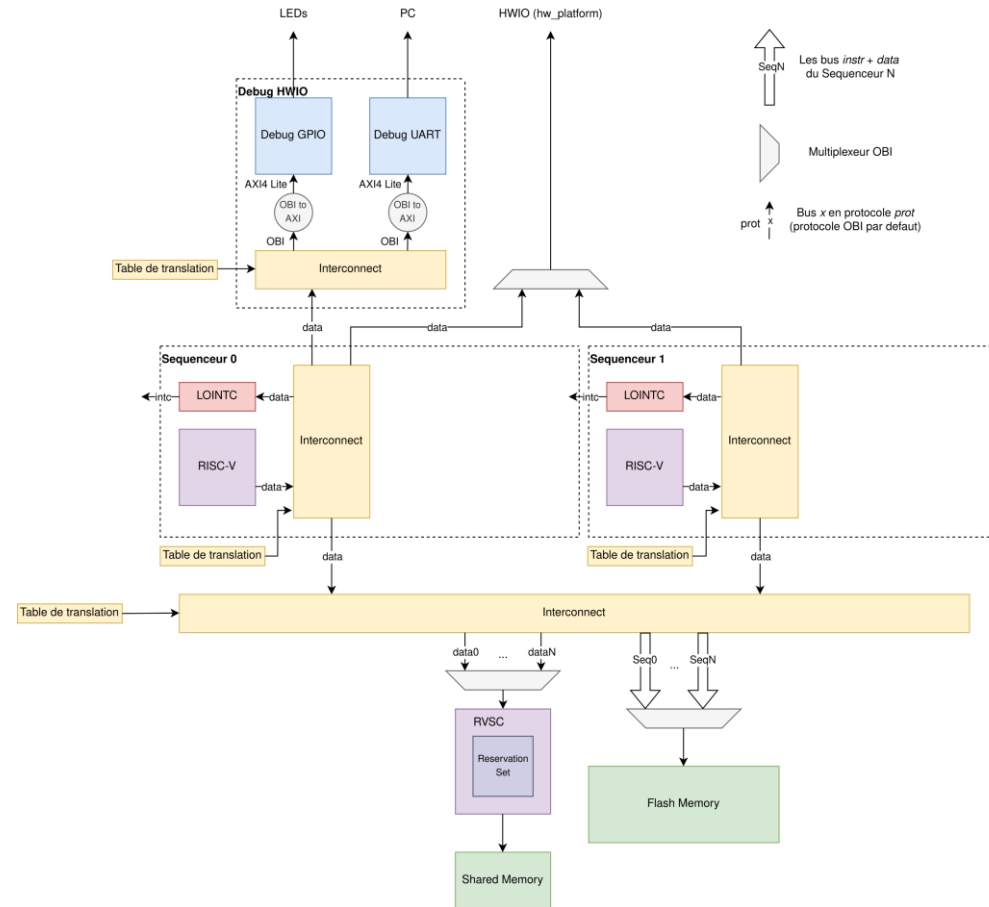


Architecture cosimulée

- Les RISC-V sont émulés donc leur instantiation VHDL a été commentée.

- IPs non utilisées lors de la cosimulation:

- CPUs
- Ips de débogage et bus associés (sba)
- Mémoires non partagées (émulées dans QEMU)
- Bus d'instructions



A stylized profile of a human head facing right, filled with a blue-to-white gradient. Inside the head, a network diagram is visible, consisting of numerous white dots connected by thin white lines, representing a complex system or data flow. The background of the slide is a light blue gradient.

Application au test d'IPs et de drivers sur NG-ULTRA

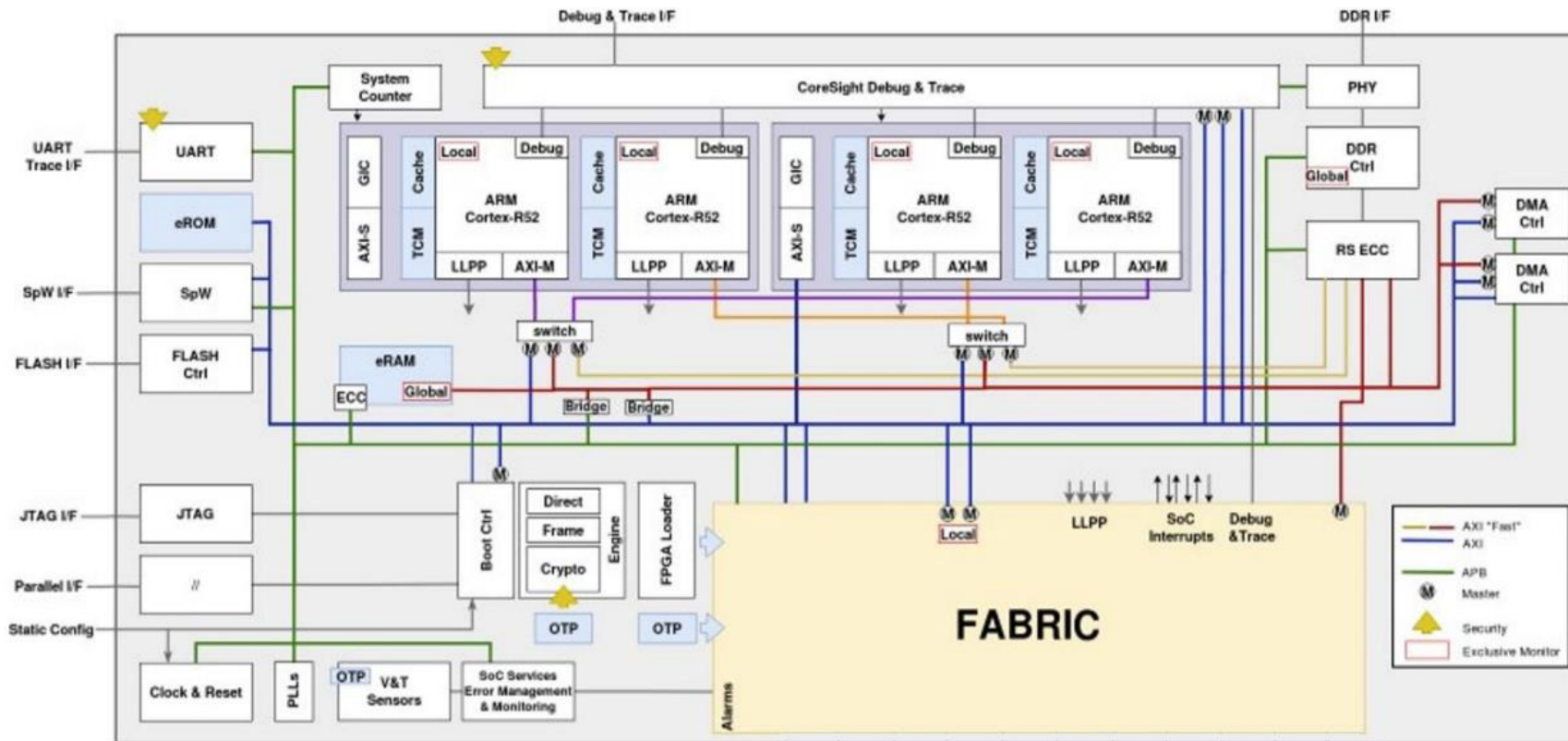
► Implémentation machine QEMU

- Différent du projet multi RISC-V car il y a une partie PS donc:
 - Il y a des périphériques (interconnect, mémoires, ...) dont on a pas le source HDL
 - Le FPGA reçoit les accès des CPUs par un même bus.

➡ La machine QEMU représente le SoC entier:

- Elle contient les 4 cœurs au lieu d'un seul -> une seule instance de QEMU pour tous les cœurs.
- Le FPGA devient un périphérique comme les autres.
- Elle a une topologie complexe (mémoires locales à chaque cœurs, ...)

Architecture NG-ULTRA



Seuls les périphériques nécessaires au boot et à la communication avec la partie PL sont implémentés dans QEMU.

▶ Prochaines étapes

- Support du GIC sur NG-ULTRA afin de bien réceptionner les interruptions générées par le FPGA
- Possible amélioration de la cosimulation multi RISC-V avec l'utilisation d'une seule instance de QEMU pour gérer tous les cpus.



FIN