AFF3CT et RISC-V

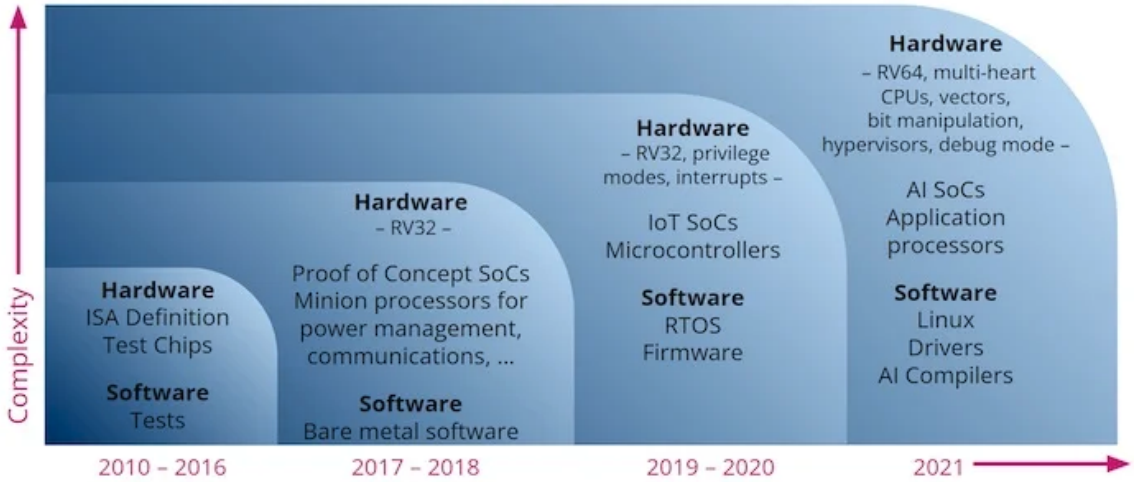
Perspectives de recherche

Camille Leroux, Mathieu Escouteloup

Journée AFF3CT (2ème édition)

LIP6, Paris, 28 Novembre 2023

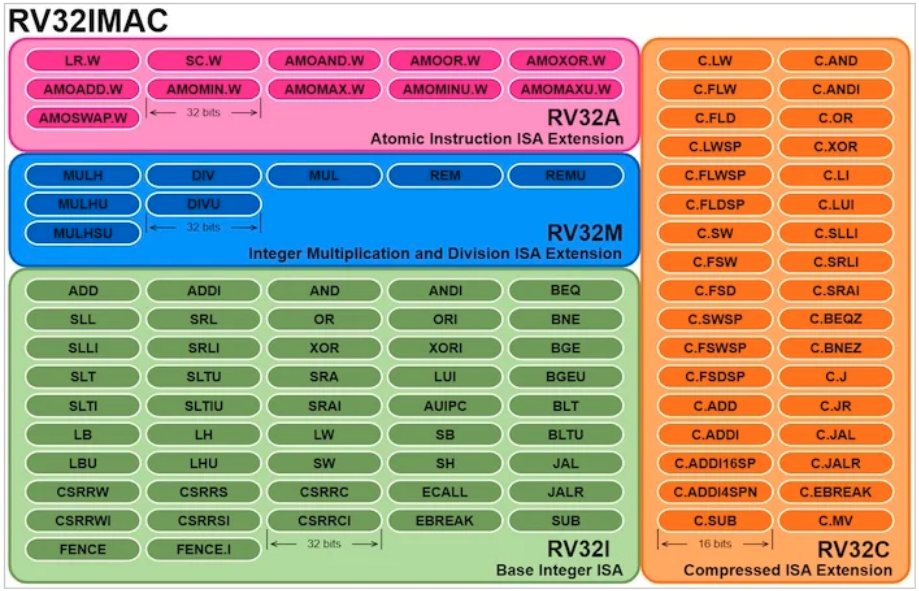
RISC-V Timeline



RISC-V = Jeu d'instructions extensible et open source

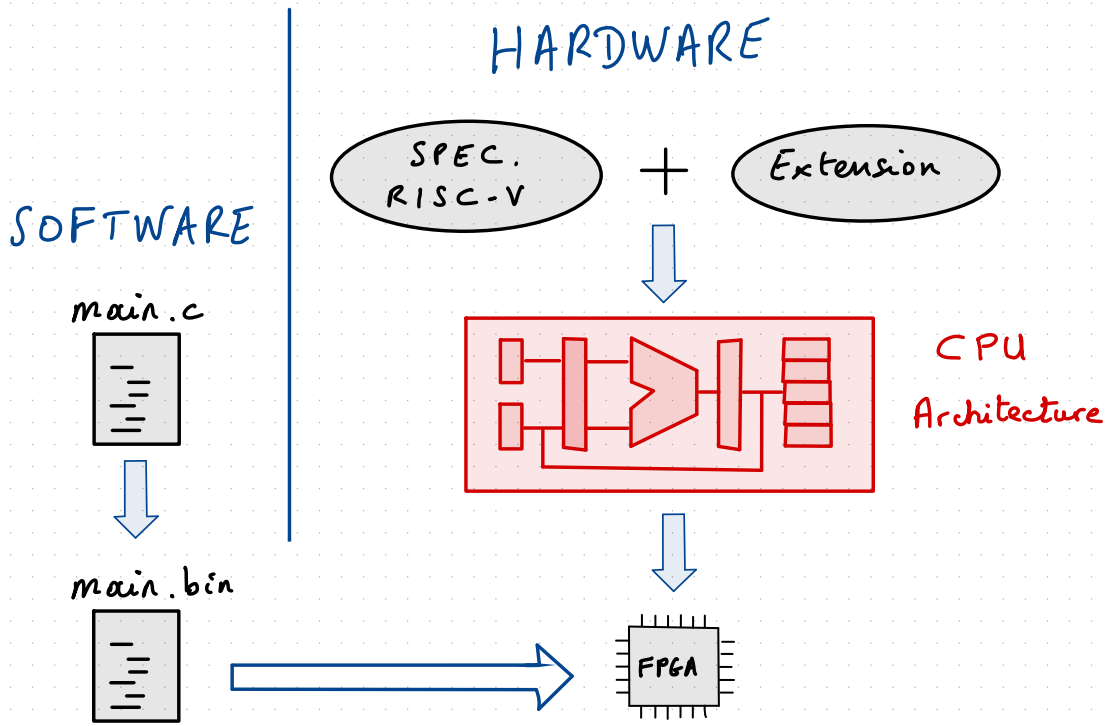
- 2010 : Projet de fin d'étude (3 mois) à UC Berkeley
- 2015 : RISC-V est rendu public / Création de la RISC-V foundation
- 2018 : Collaboration avec Linux foundation

Source : riscv.org



- Convention de nomenclature : RV + bit width + extension ID
- Exemple : RV32-IMAC :
  - RV32I : CPU 32-bit avec l'ISA de base **I**nteger
  - M : **M**ultiplication et Division
  - A : Instructions **A**tomiques
  - C : Instructions **C**ompressées

RISC-V : Design flow



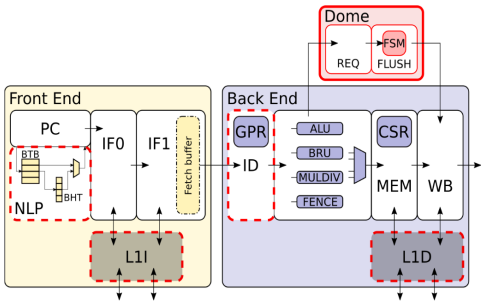
- Customizable hardware
- Customizable tool chain

Pourquoi RISC-V ?

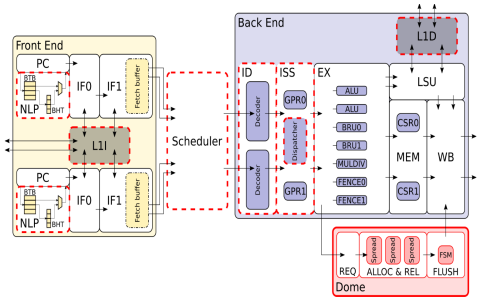
- ISA flexible et adaptable au besoin (extensions, customisation), coproc., ...)
- Architecture prototypable sur FPGA
- Projet open source d'envergure mondiale `https://riscv.org`
- Expertise conception HW/SW présente dans l'équipe : (M. Escouteloup, J. Saussereau, F. Pouget, M. Legras-Chevalier)

Pourquoi RISC-V ?

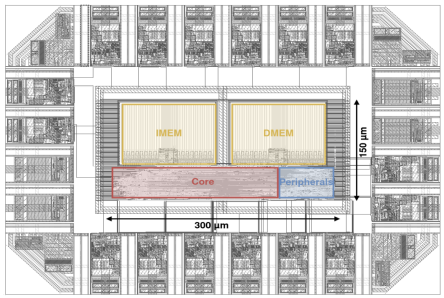
- ISA flexible et adaptable au besoin (extensions, customisation), coproc., ...)
- Architecture prototypable sur FPGA
- Projet open source d'envergure mondiale `https://riscv.org`
- Expertise conception HW/SW présente dans l'équipe :
  (M. Escouteloup, J. Saussereau, F. Pouget, M. Legras-Chevalier)



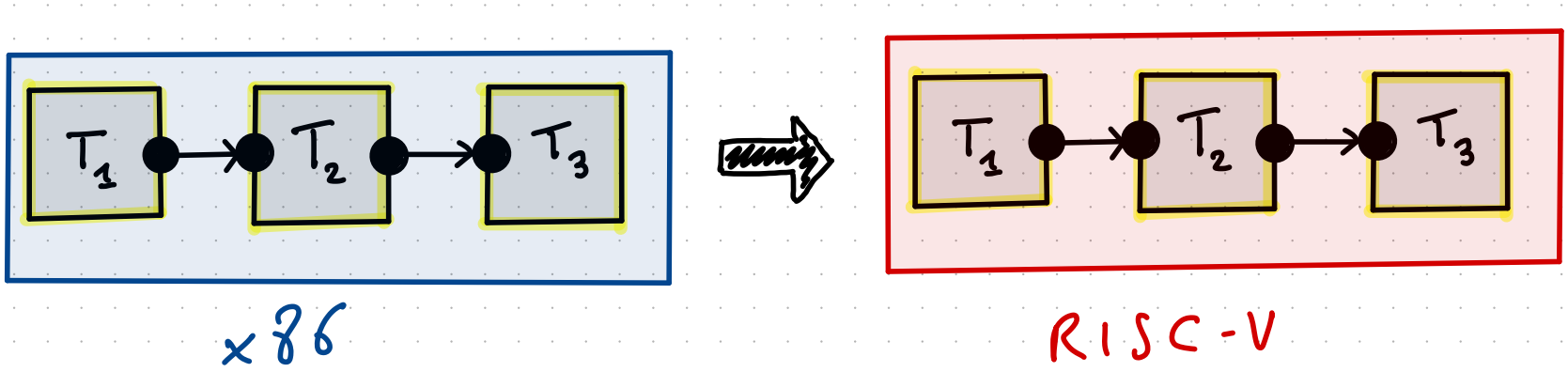
AUBRAC
RV32IM + ZiCSR + Zifencei



SALERS
RV32IM + ZiCSR + Zifencei
Superscalaire + SMT



ASTERISC
RV32I(MC)



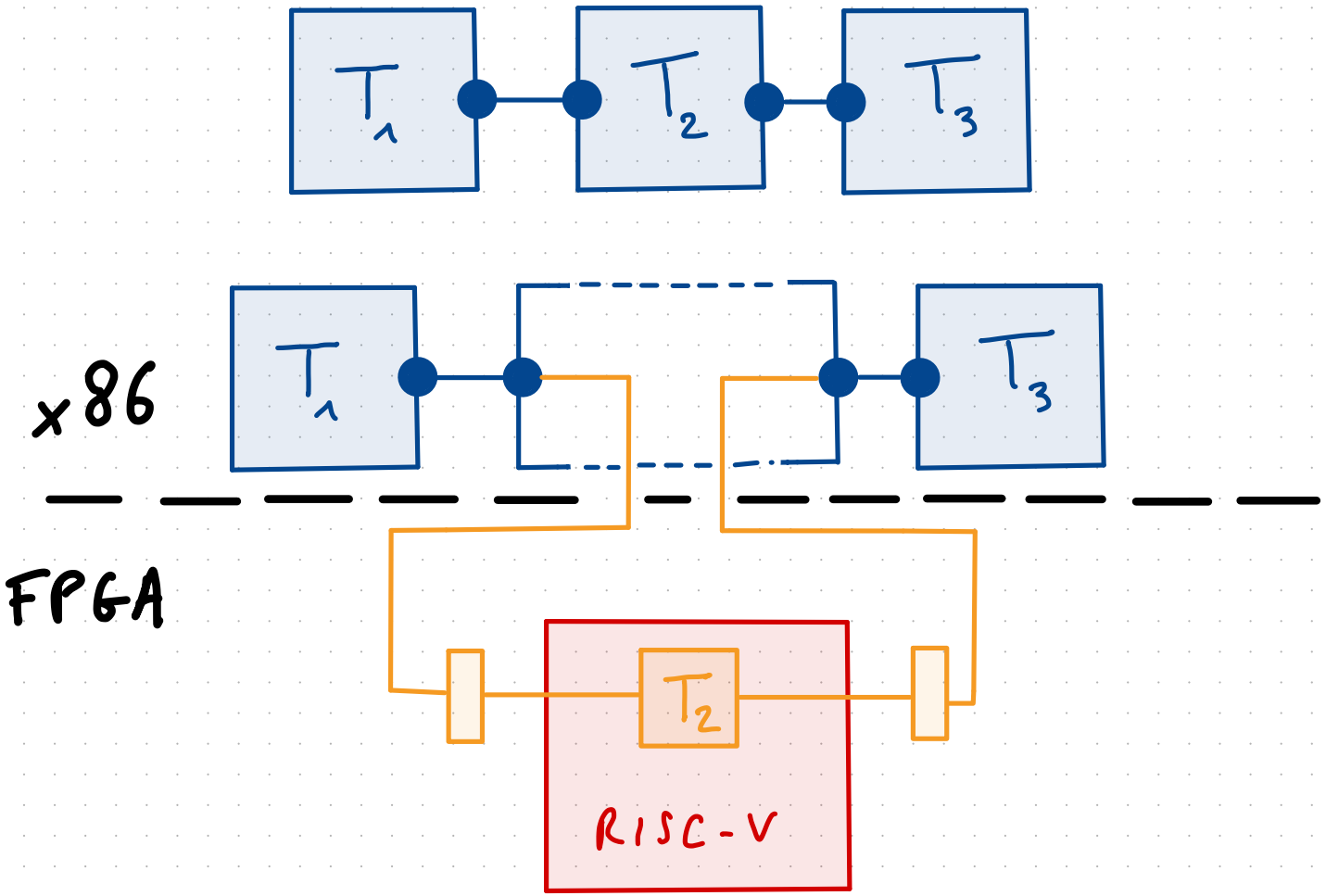
Intérêts

- Evaluer l'ISA RV
- Adapter l'extension RV
- Dimensionner la mémoire
- Conserver la flexibilité du SW
- Comparer avec ARM/x86/...

Défis

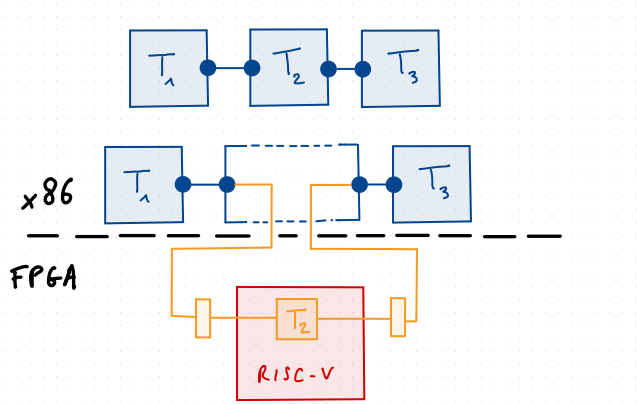
- Compiler AFF3CT pour RV
- Estimer les perf. des tâches avant implémentation
- Occupation (FPGA) si chaîne complexe

RISC-V in the (AFF3CT) loop



Intérêts

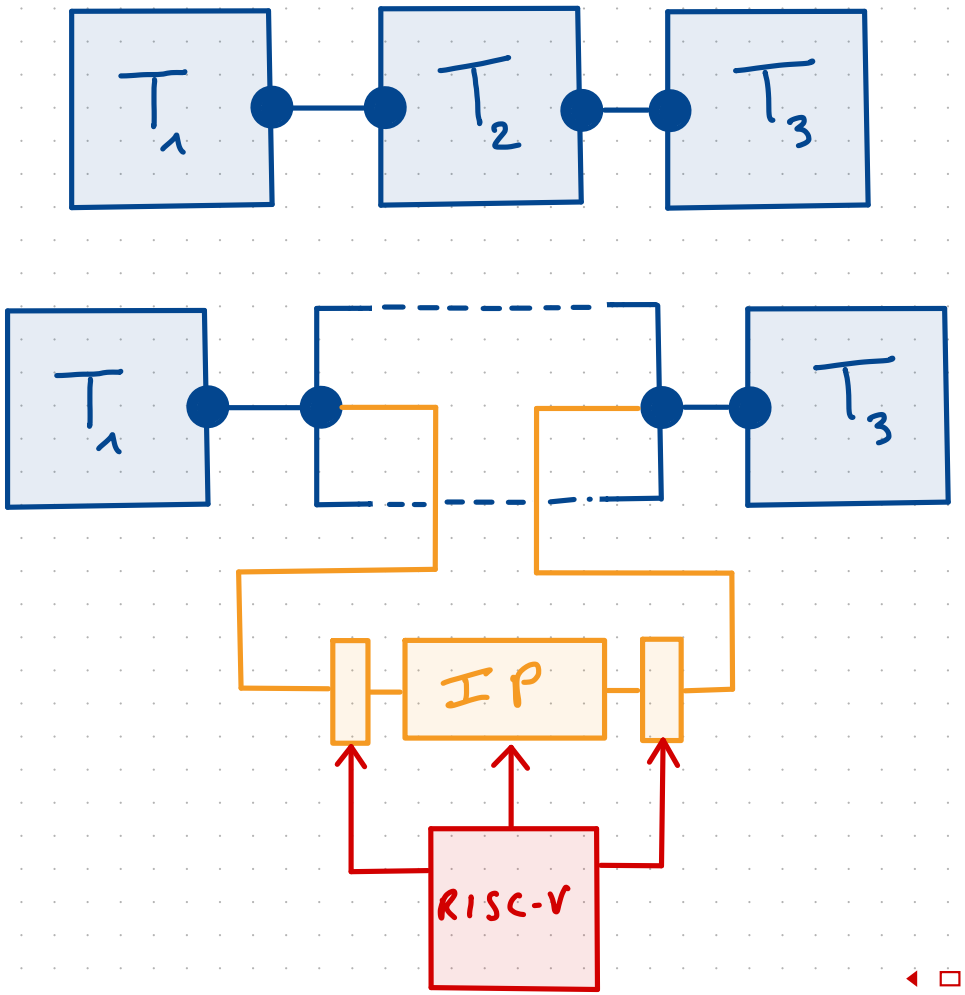
- Evaluer un sous-ensemble de tâches
- Utiliser x86 pour simuler l'environnement
- Portable sur n'importe quel FPGA
- Spécialiser le RV en fonction des tâches



Défis

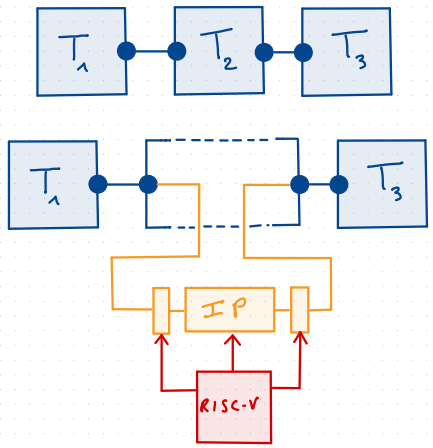
- Concevoir des sockets HW flexibles et génériques
- Concevoir des interfaces logicielles (R/W)

Hardware in the (AFF3C) loop



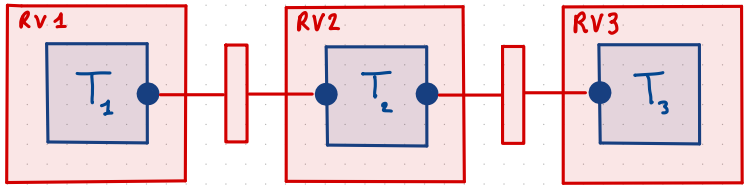
Intérêts

- Disposer d'une interface générique pour tester une IP
- Evaluer les performances de l'IP
- RV faible complexité devrait être suffisant



Défis

- Contrôle du flux de données (DMA, RT, FIFO, ...)



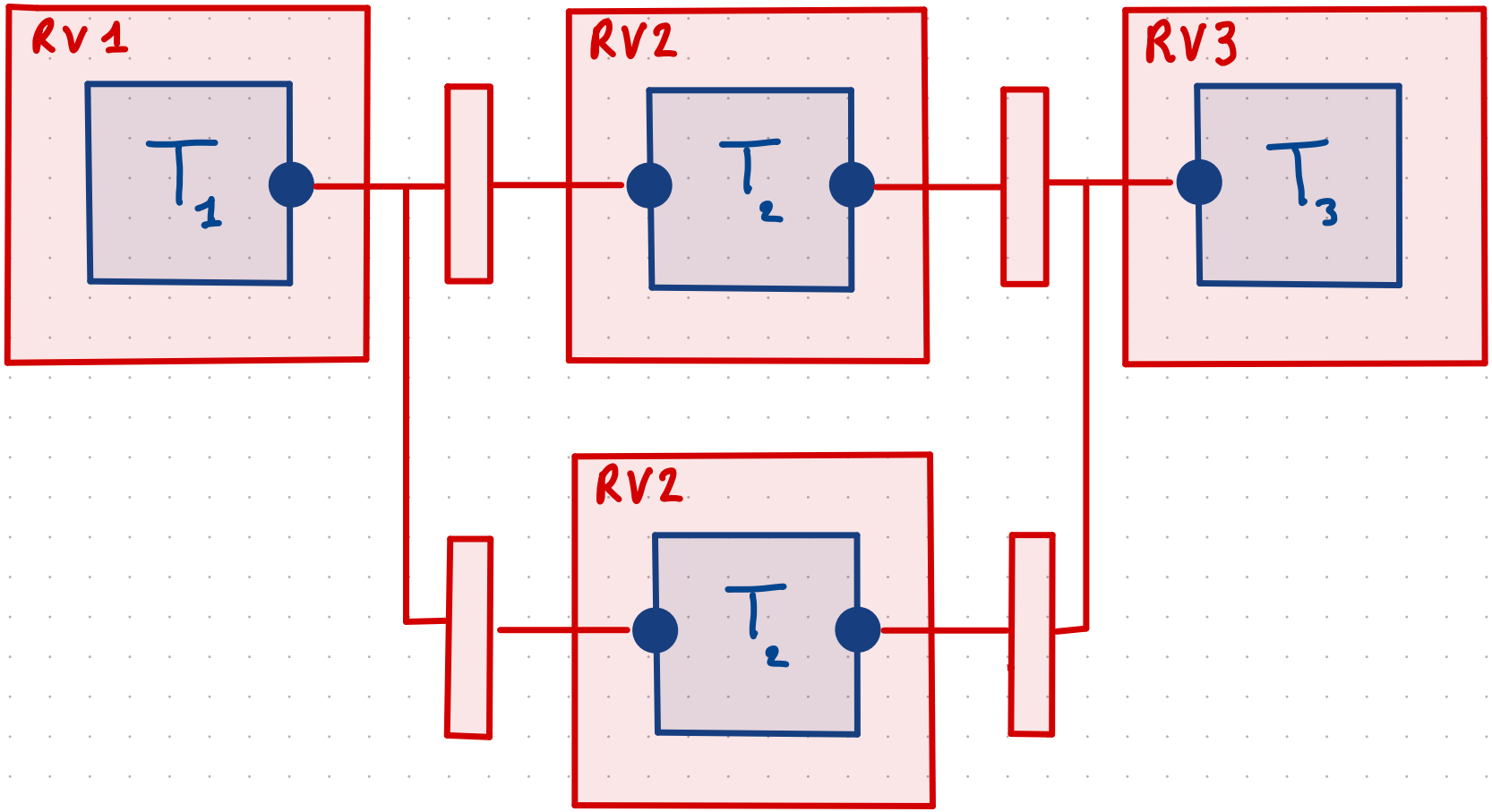
Intérêts

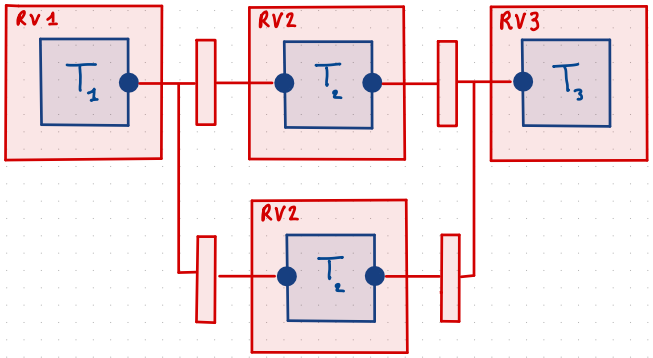
- Disposer d'un pipeline logiciel pour l'embarqué
- Chaque RV peut avoir une archi. adaptée à sa tâche

Défis

- Définir les sockets (mém. partagée, DMA, HW, SW, ...)
- Trouver une bonne adéquation algorithme / architecture
- Synchroniser / superviser l'exécution (coeur maitre ?, exec. implicite ?)

Architecture multi-coeurs RISC-V





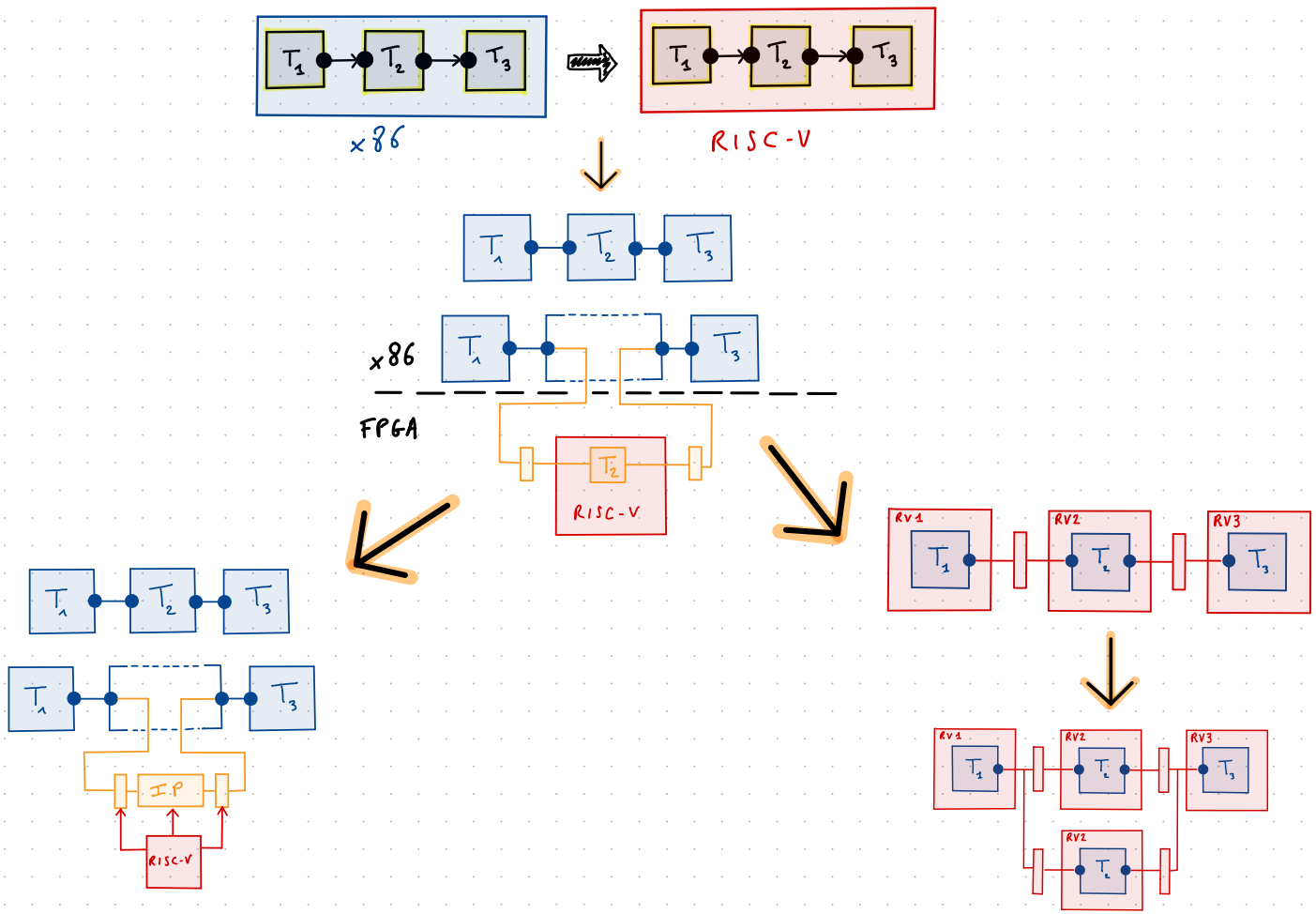
Intérêts

- Adapter le type de parallélisme à l'application
- Plateforme multi-coeur hétérogène flexible faible complexité

Défis

- Estimer la durée d'exécution des tâches
- Trouver la configuration optimale des coeurs
- Interconnexions ? NoC ? Bus partagé ? Bus distincts ?
- Réseau reconfigurable ?

En résumé...



That's all folks...

MERCI