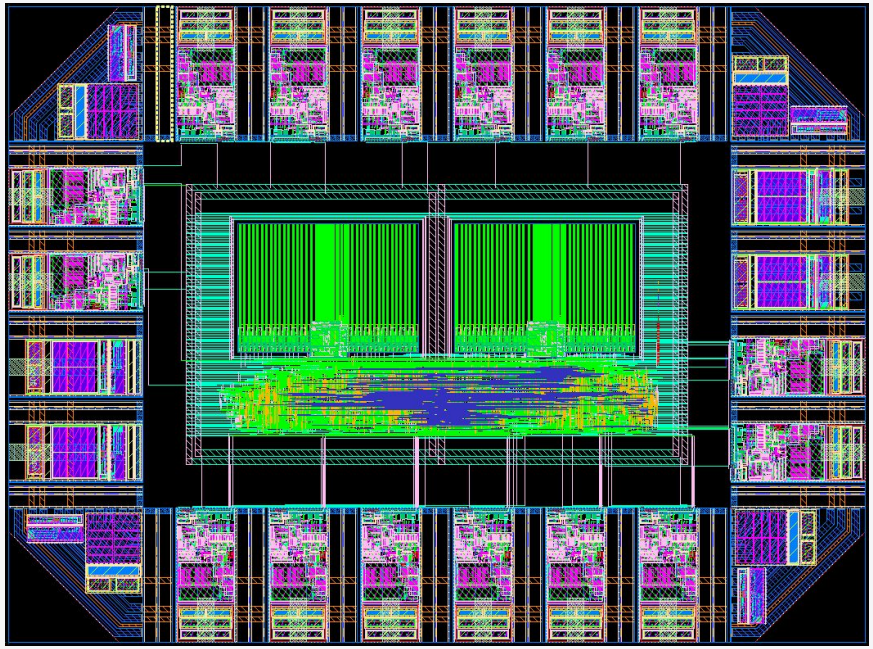


AsteRISC

Un cœur RISC-V flexible

Jonathan Saussereau
Christophe Jégo
Camille Leroux
Jean-Baptiste Bégueret

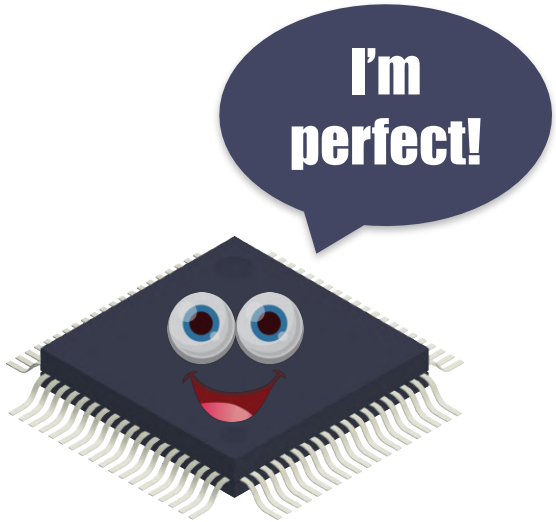
Université de Bordeaux, Bordeaux INP,
Laboratoire IMS, UMR CNRS 5218

{prénom.nom}@ims-bordeaux.fr

Sommaire

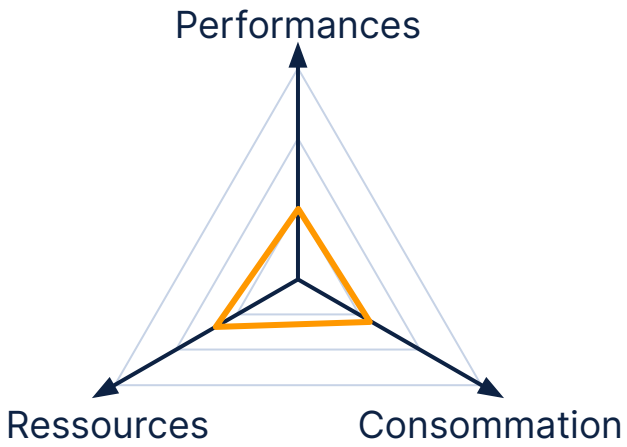
Contexte

Il n’y a pas de CPU parfait !



Contexte

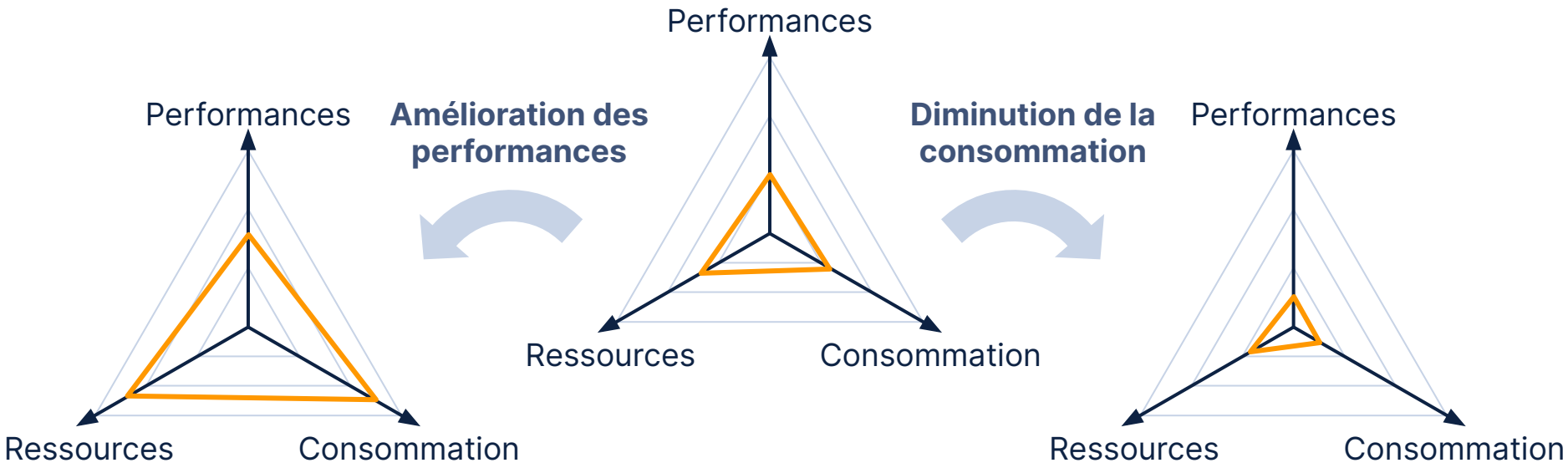
Compromis dans la conception de CPUs



- Chaque architecture a un compromis entre
  - ➜ Performances
  - ➜ Utilisation des ressources (coût)
  - ➜ Consommation énergétique
- Certains compromis sont plus adaptés à certains cadres applicatifs que d'autres

Contexte

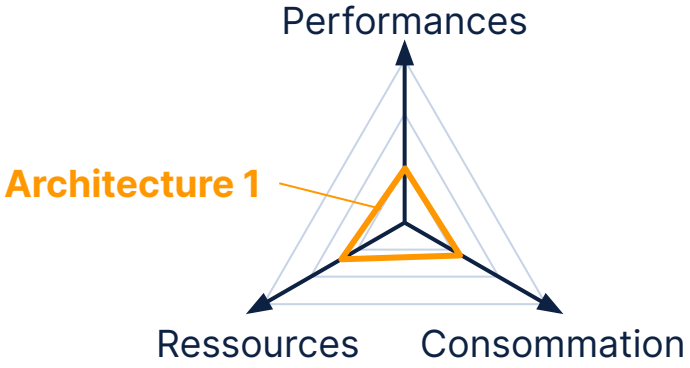
Compromis dans la conception de CPUs

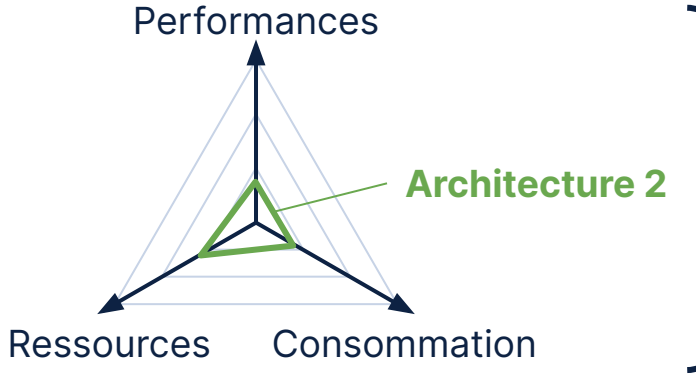


➜ Chercher à modifier une métrique impactera les autres

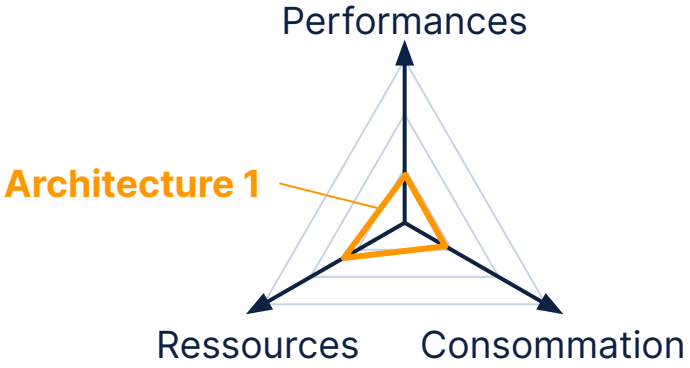
Contexte

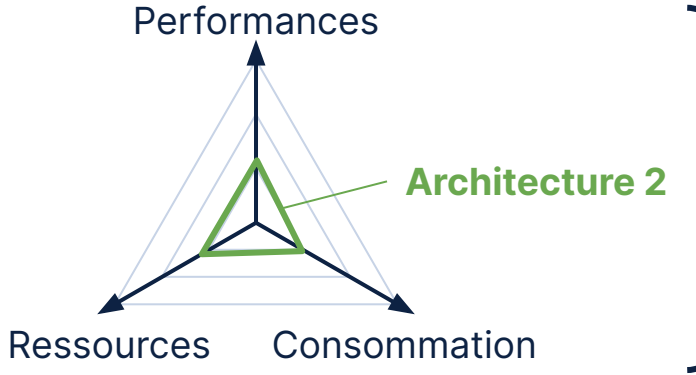
Changer de cible change le compromis





Cible technologique **A**

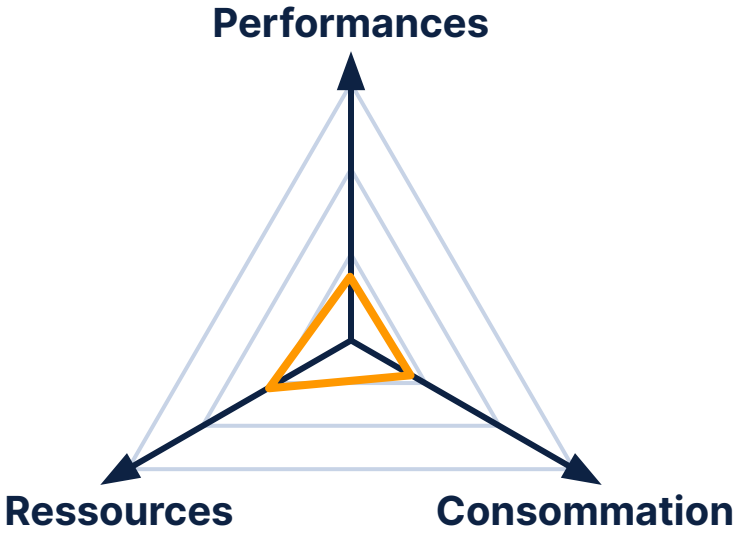




Cible technologique **B**

Contexte

Le compromis d'un CPU n'est pas absolu



Le compromis est différent pour chaque cible technologique (modèle de FPGA, technologie ASIC)

Les performances du CPU dépendent de ce qu'il y a autour (mémoire, périphériques, ...)

Un CPU flexible permettrait de chercher le meilleur compromis pour un cadre applicatif donné

Première approche

Une architecture non-pipeline flexible

Contexte et besoin

. . .

Approche non-pipeline

. . .

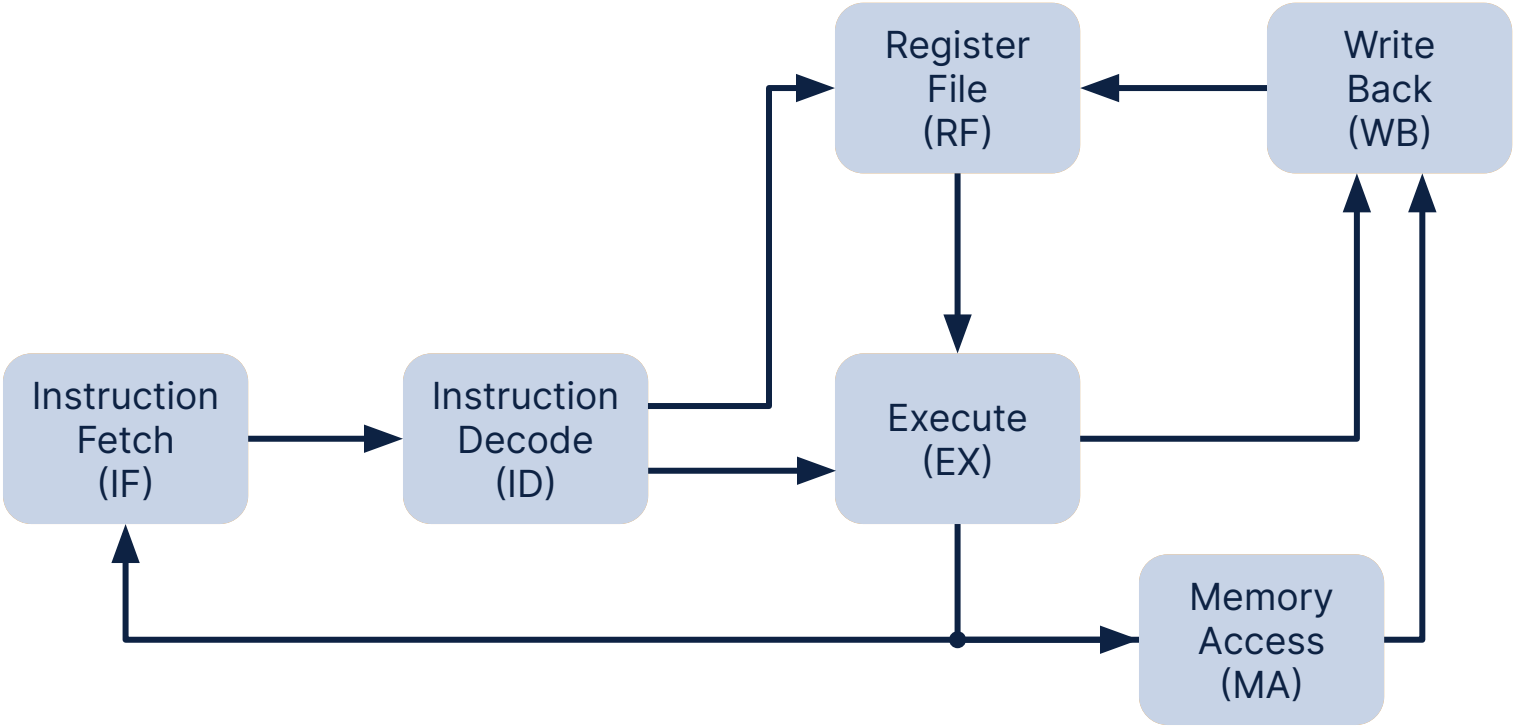
Approche pipeline

. . .

Résultats obtenus

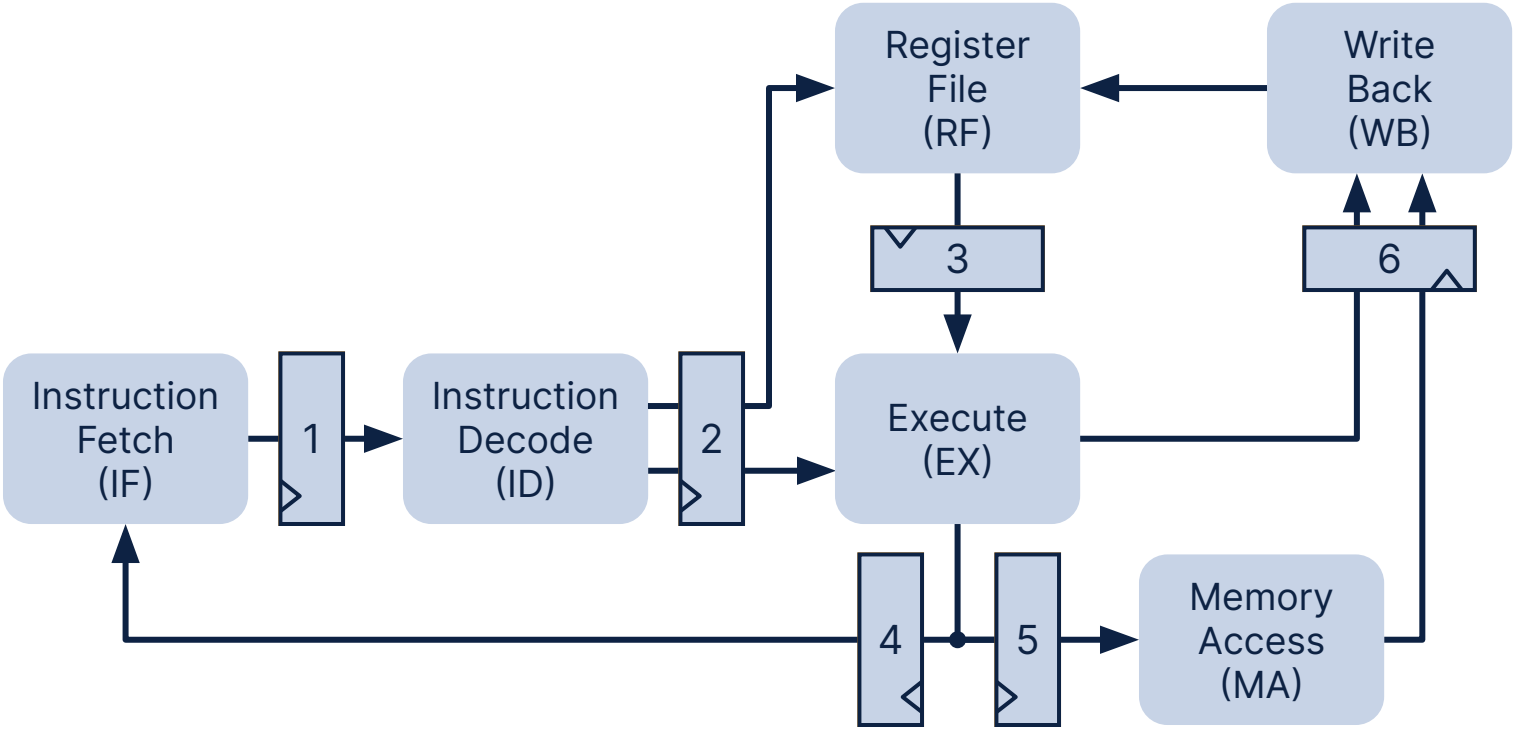
Approche non-pipeline

Schéma d'un processeur RISC classique



Approche non-pipeline

Schéma de l'architecture flexible proposée



Approche non-pipeline

Principes mis en place pour AsteRISC

Registres optionnels à des points critiques du datapath

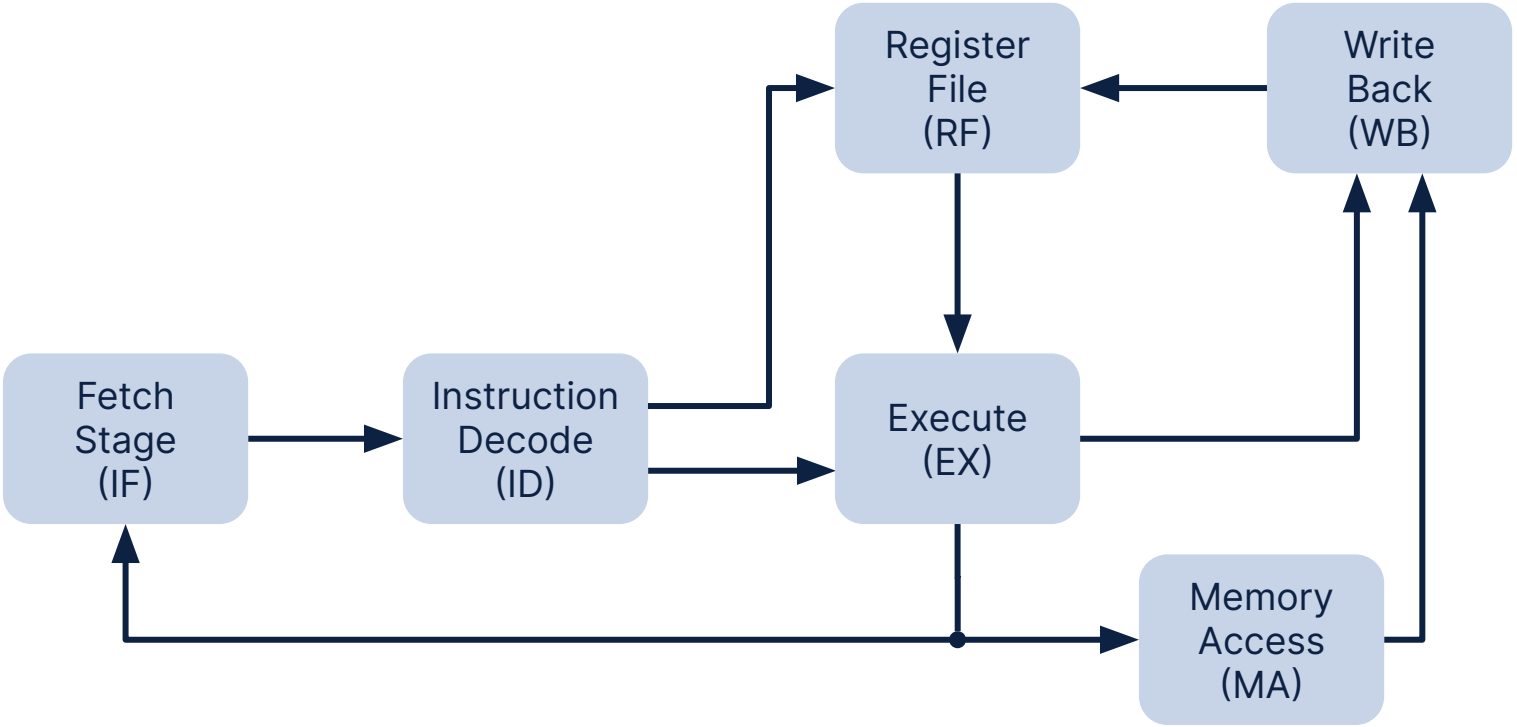
Unité de contrôle à base de machine d' état flexible

Description en SystemVerilog pour faciliter sa compréhension et sa modification

La flexibilité se situe avant synthèse. Après synthèse, l'architecture est fixe et il n'y a pas de surcoût.

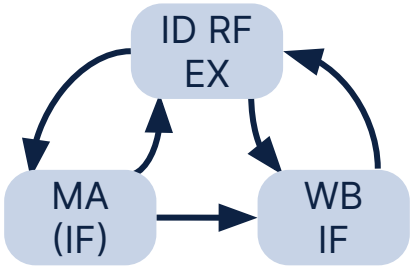
Approche non-pipeline

`M0000` : Configuration architecturale de base

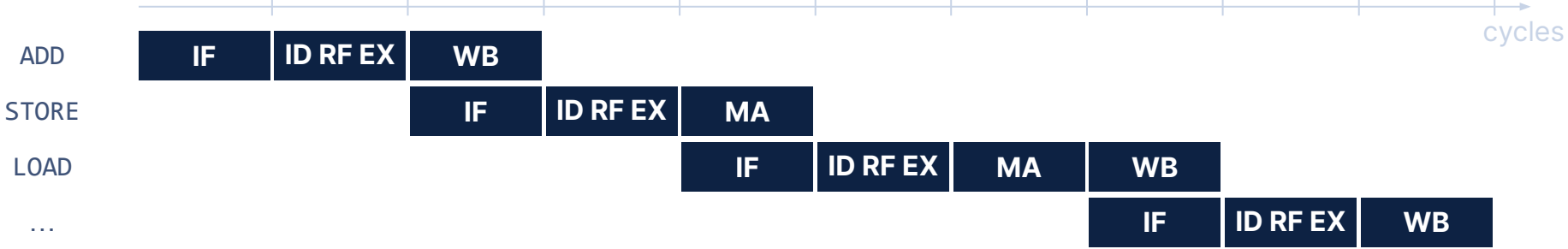


Approche non-pipeline

M0000 : Configuration architecturale de base

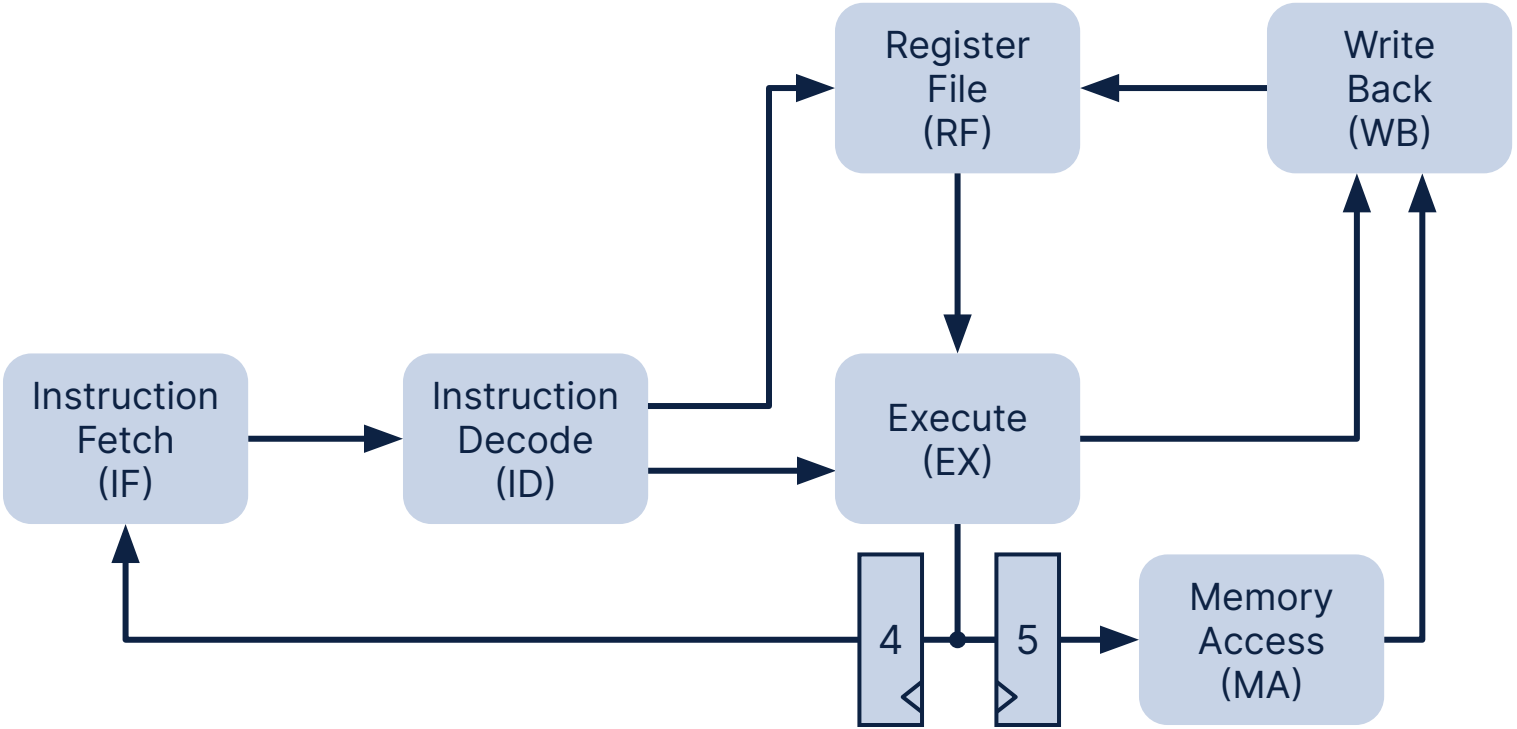


- Pour les applications à basse fréquence
- Aucun registre optionnel n'est activé
  - ➜ 3 cycles pour les instructions load
  - ➜ 2 cycles pour les autres instructions



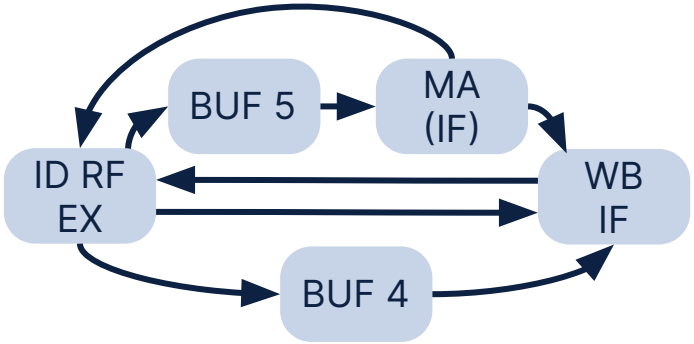
Approche non-pipeline

M0024 : Bufferisation des accès mémoire

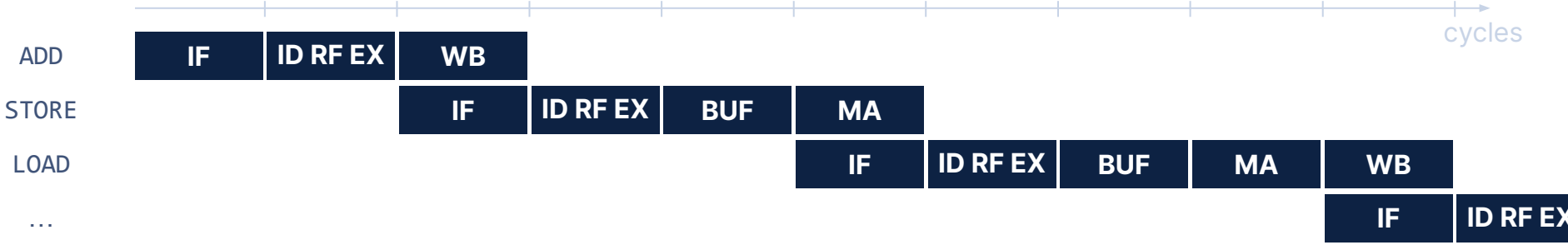


Approche non-pipeline

M0024 : Bufferisation des accès mémoire

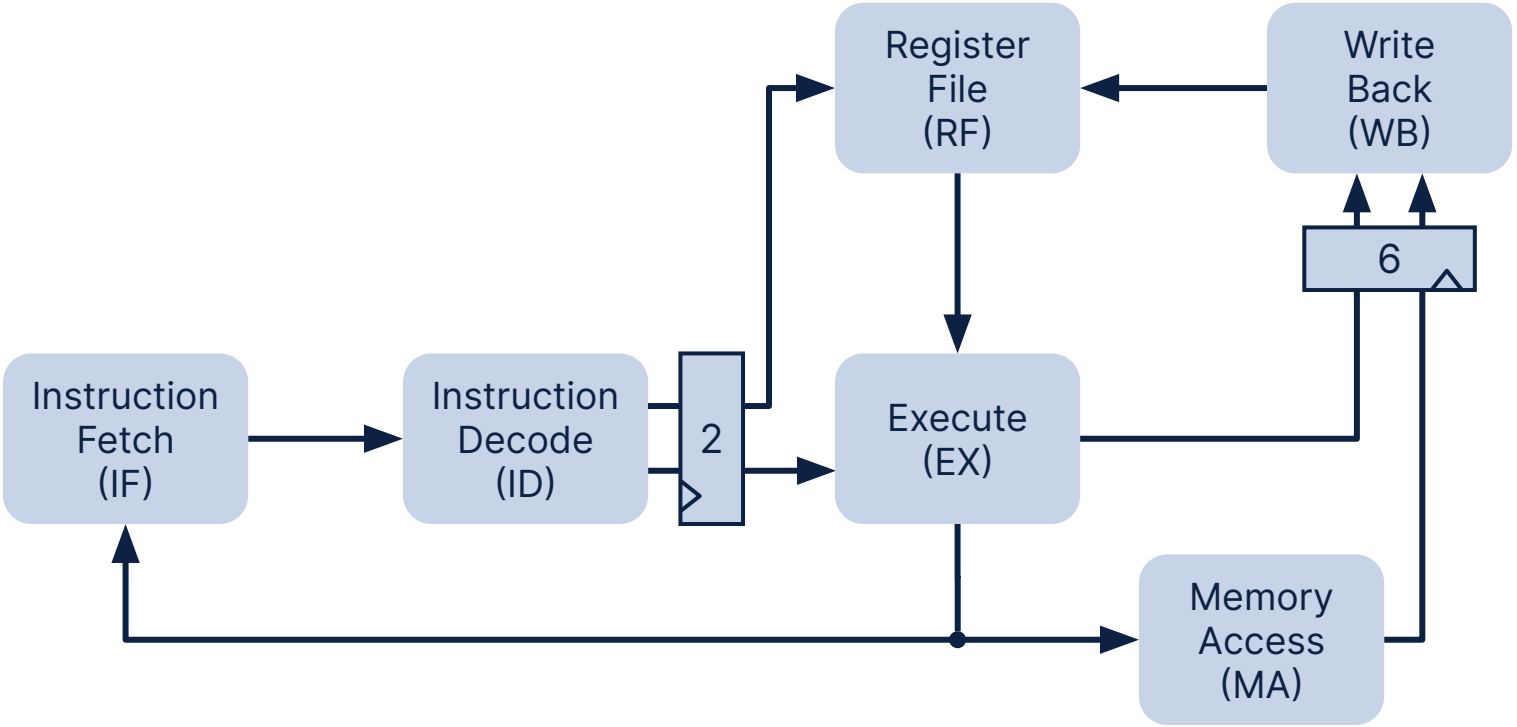


- Dans le cas de mémoires lentes
- Bufferisation des accès mémoire
  - ➜ 4 cycles pour les instructions load
  - ➜ 3 cycles pour les instructions store
  - ➜ 3 cycles pour les sauts
  - ➜ 2 cycles pour les autres instructions



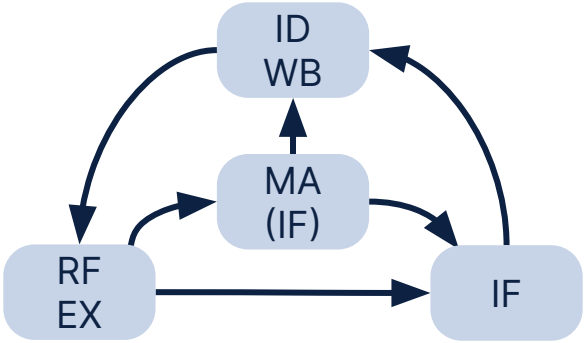
Approche non-pipeline

M0036 : Séparation d'étages



Approche non-pipeline

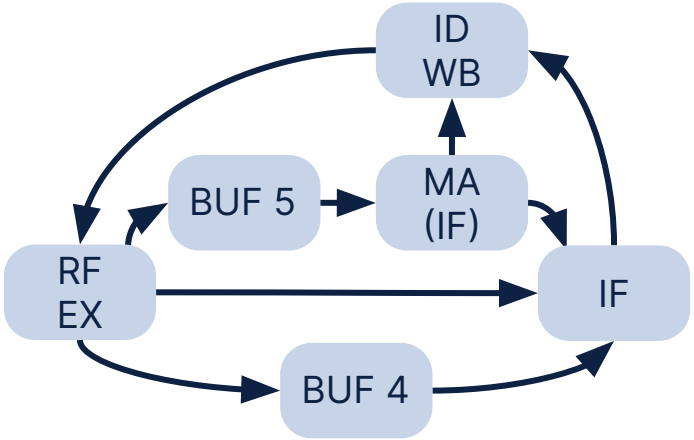
M0036 : Séparation d'étages



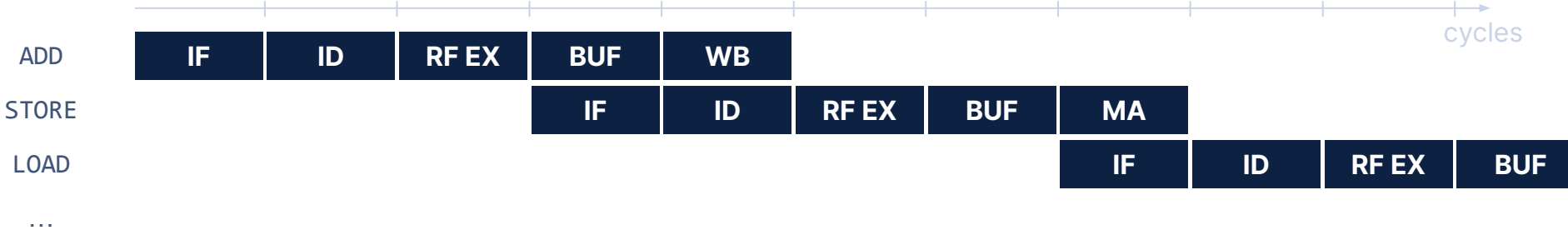
- Dans le cas de mémoires rapides
- Séparation entre ID et RF+EX
  - ➜ 4 cycles pour les instructions load
  - ➜ 3 cycles pour les autres instructions

Approche non-pipeline

M0060 : Synthèse des propositions précédentes



- Minimisation du chemin critique
- Synthèse des propositions précédentes
  - ➜ 5 cycles pour les instructions load
  - ➜ 4 cycles pour les instructions store
  - ➜ 4 cycles pour les sauts
  - ➜ 3 cycles pour les autres instructions



Approche non-pipeline

Conclusion sur l'approche non-pipeline

AsteRISC
non pipeline

- Architecture non pipeline
- Registres optionnels entre étages
- Unité de contrôle flexible à base de FSM
- L'ajout de registres tend à augmenter f_{max}
- L'ajout de registres augmente le CPI

Deuxième approche
Une architecture pipeline flexible

Contexte et besoin

Approche non-pipeline

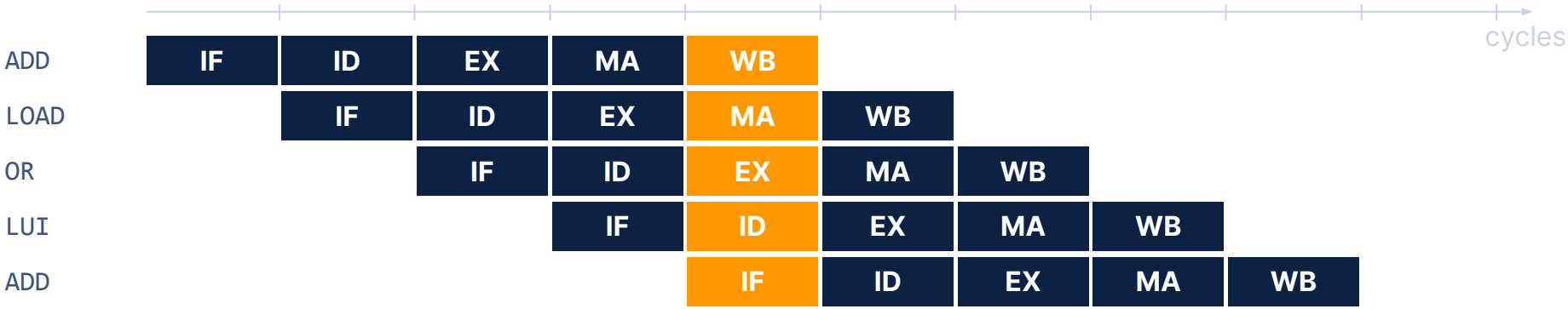
Approche pipeline

Résultats obtenus

Approche pipeline

Principe du pipeline

- Pipeline à 5 étages



- Principe du pipeline :
  - ➜ Plusieurs instructions sont traitées en mêmes temps
  - ➜ Tous les étages sont utilisés en même temps
  - ➜ Nombre de cycles par instruction proche de 1

Approche pipeline

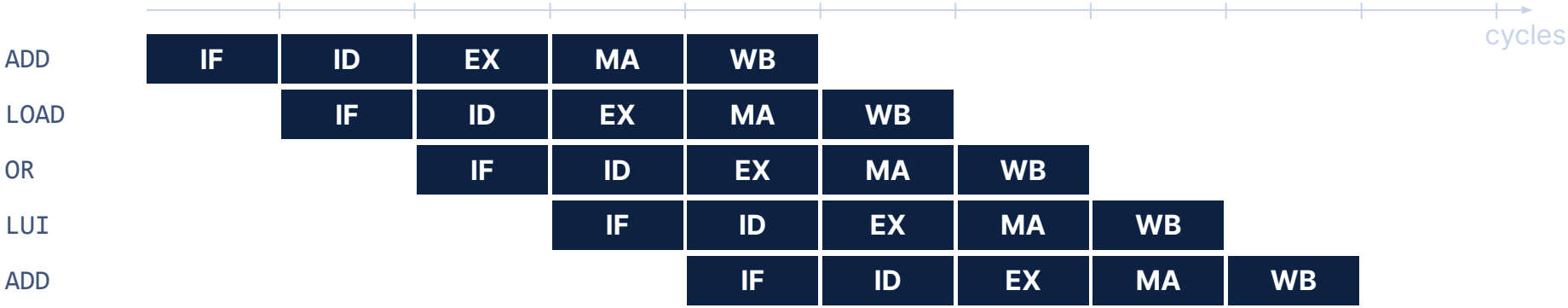
Pipeline à nombre d'étage flexible

- Pipeline à 4 étages

Approche pipeline

Pipeline à nombre d'étage flexible

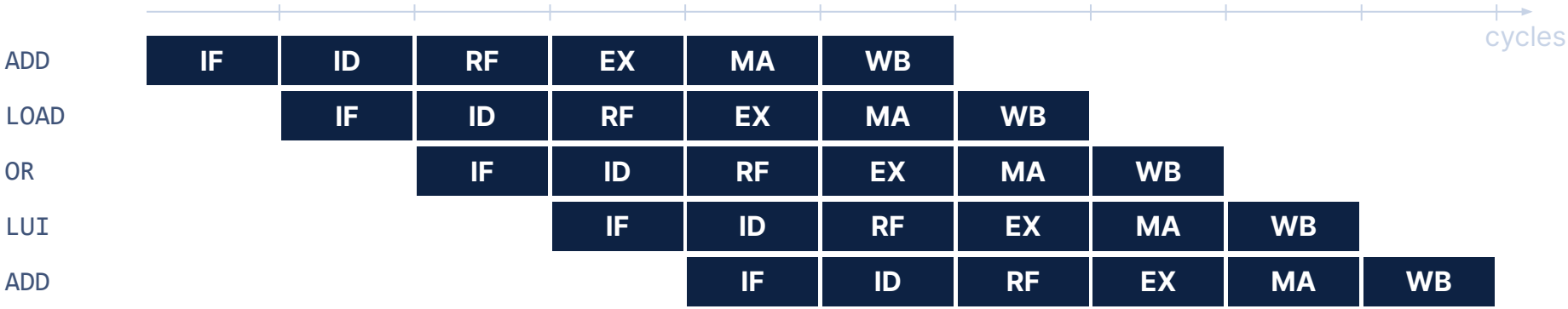
- Pipeline à 5 étages



Approche pipeline

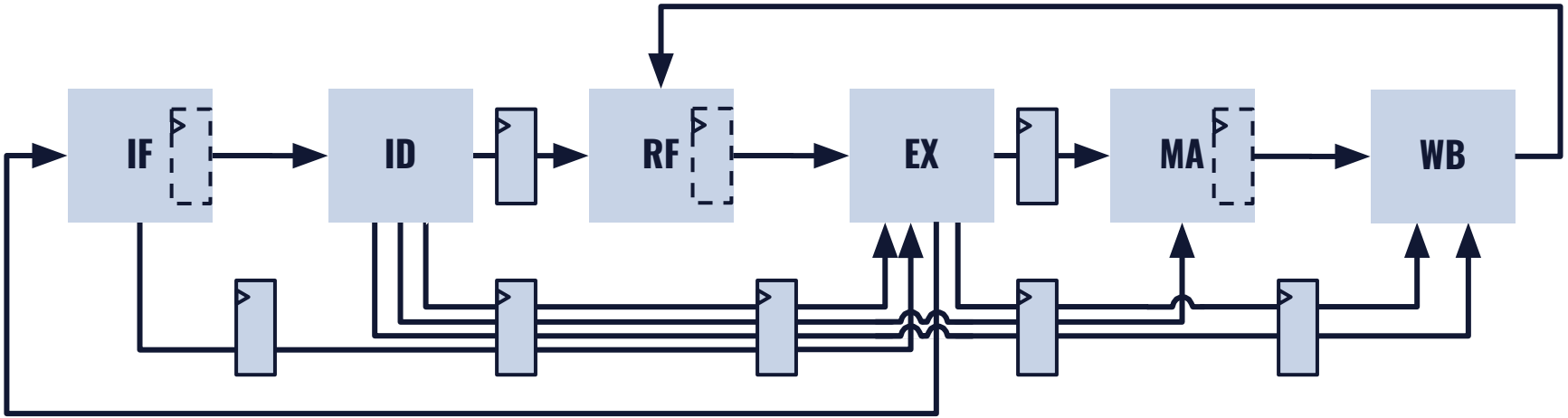
Pipeline à nombre d'étage flexible

- Pipeline à 6 étages



Approche pipeline

Schéma de l'architecture flexible proposée



- Pipeline flexible
  - ➜ Les barrières de registres peuvent être activées ou désactivées
  - ➜ On peut ainsi choisir le nombre d'étages

Approche pipeline

Barrière de registre flexible

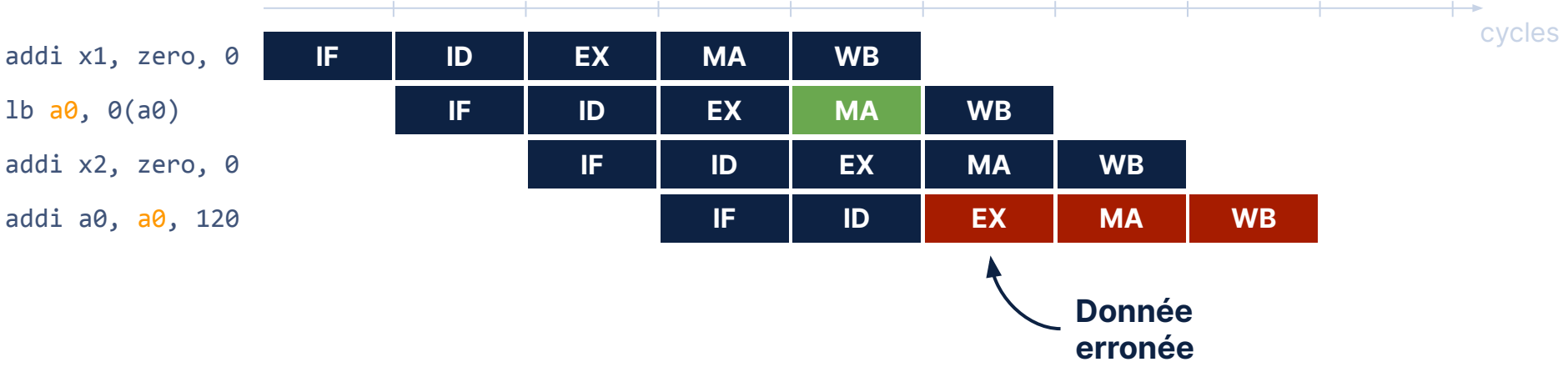
Barrière de registre flexible

```
if (p_stage_EX) begin
  always_ff @(posedge i_clk) begin
    if (i_rst) begin
      alu_out_EX      <= 32'b0;
      sel_wb_EX       <= wb_none;
    end else begin
      alu_out_EX      <= alu_out;    // Registre implémenté
      sel_wb_EX       <= sel_wb_RF;  // Registre implémenté
    end
  end
end else begin
  always_comb begin
    alu_out_EX      = alu_out;       // Simple fil
    sel_wb_EX       = sel_wb_RF;     // Simple fil
  end
end
```

Approche pipeline

Gestion des Read After Write

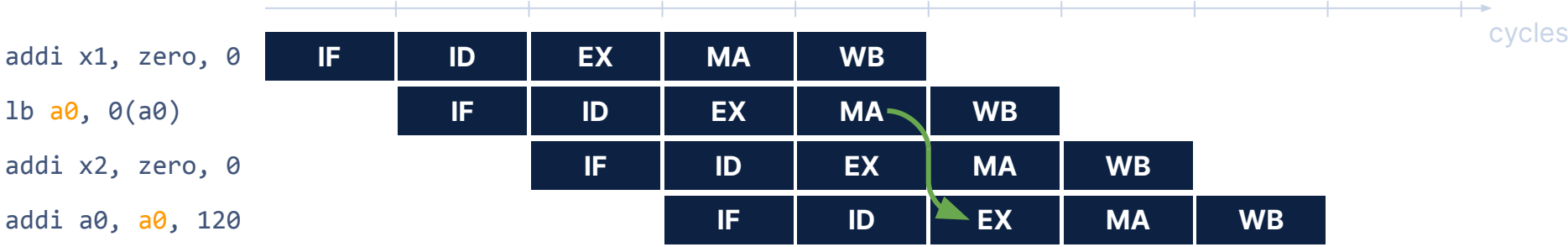
- Problème de dépendance de donnée



Approche pipeline

Gestion des Read After Write

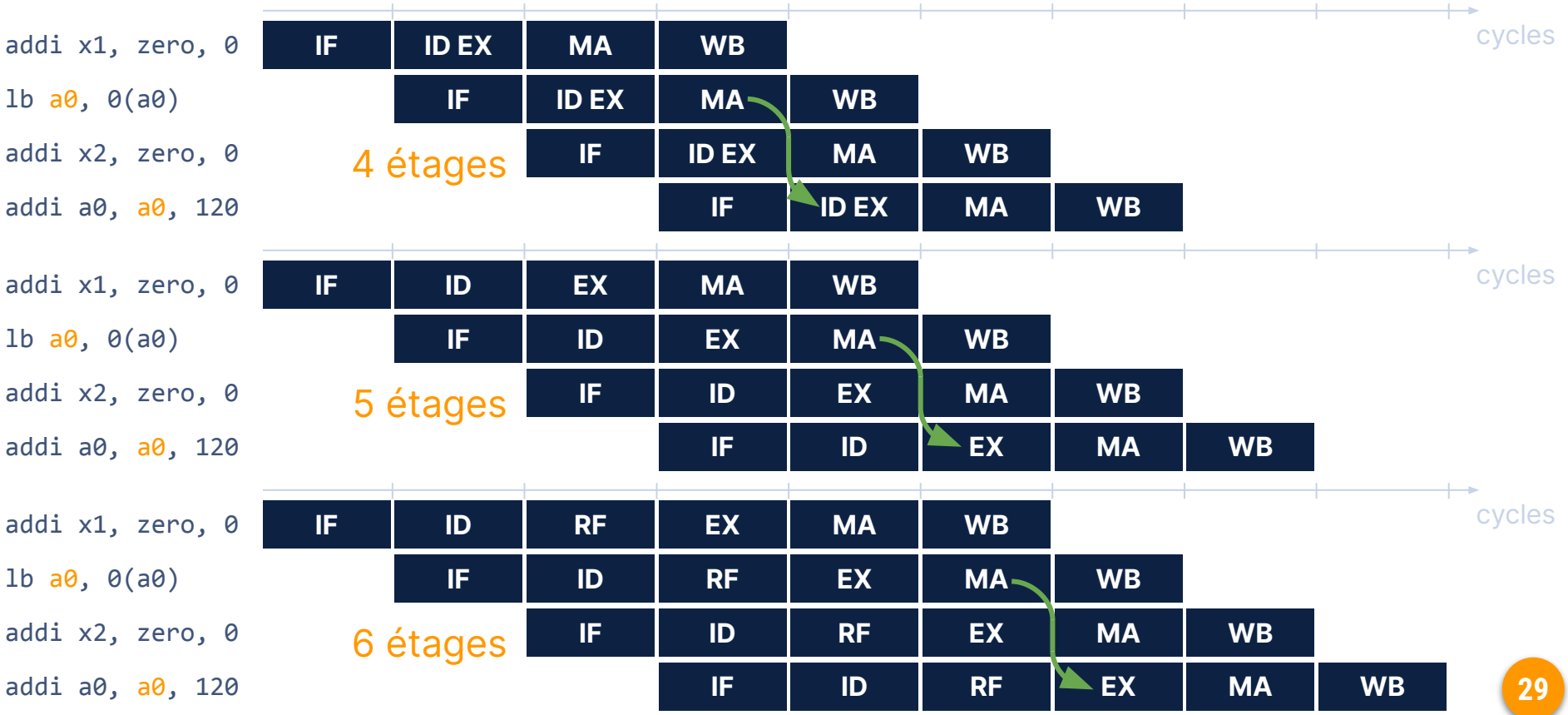
- Problème de dépendance de donnée
  - ➜ bypass entre étages



- Pipeline flexible
  - ➜ Les bypass sont souvent les mêmes dans les différentes configurations du pipeline

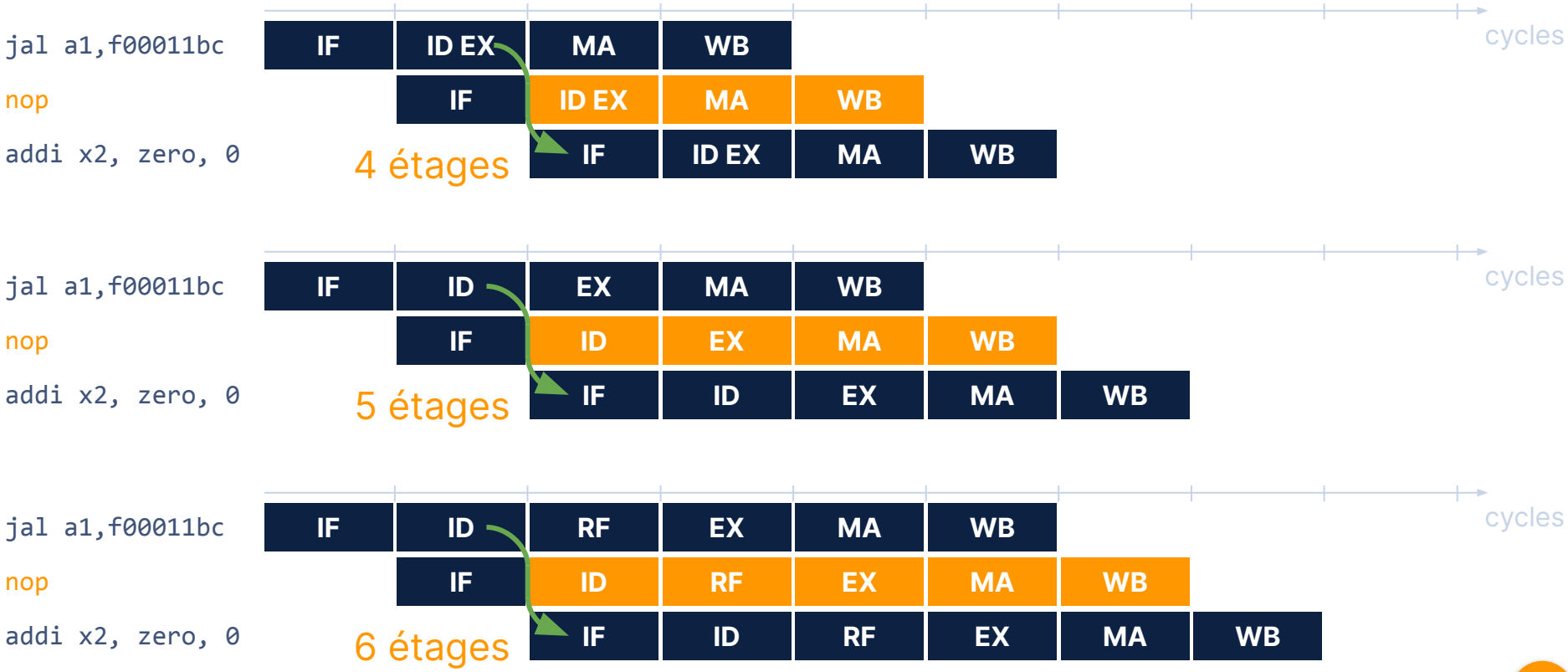
Approche pipeline

Gestion des Read After Write



Approche pipeline

Gestion des sauts inconditionnels



Approche pipeline

Conclusion sur l'approche pipeline

- Architecture pipeline
- Barrières de registres optionnelles
- Nombre d'étage au choix
- Unité de contrôle flexible
- Complexité supplémentaire est limitée

Résultats Obtenus

Implémentation sur ASIC et sur FPGA

Contexte et besoin

Approche non-pipeline

Approche pipeline

Résultats obtenus

Résultats obtenus

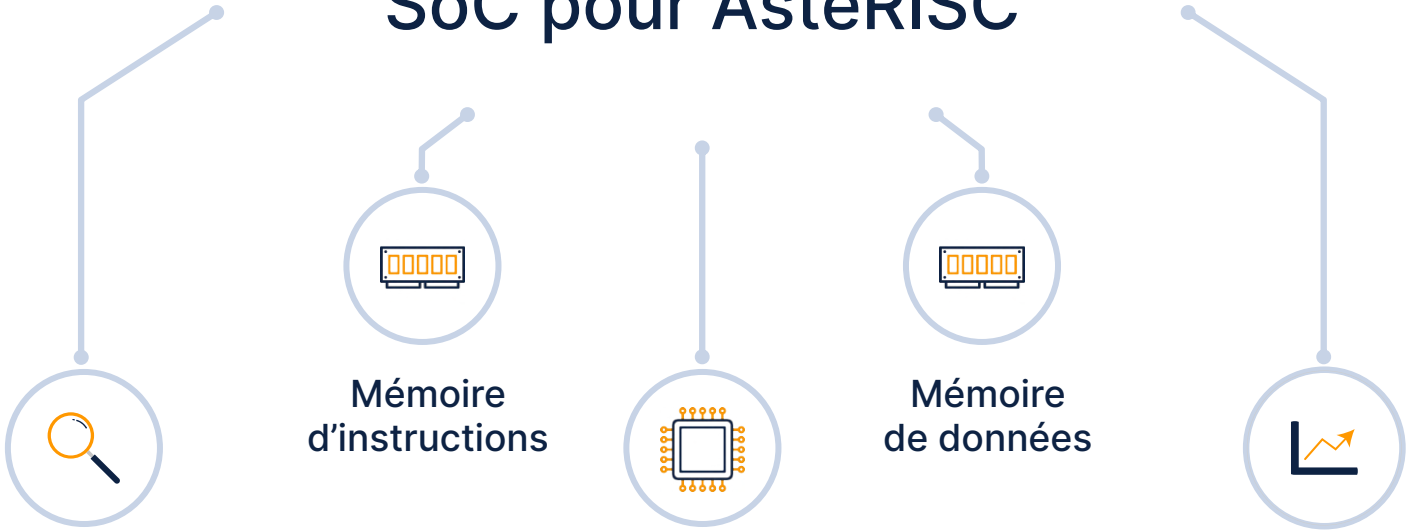
Benchmark Dhrystone

Config. d'AsteRISC	Paramètres						Benchmark		
	IF buf	IF pred	ID buf	RF buf	MA buf	WB buf	Cycles /Instruction	Dhrystones /s /MHz	DMIPS /MHz
M0000	0	0	0	0	0	0	2.151	1192	0.678
M0001	0	0	0	1	0	0	3.151	813	0.462
M0008	0	0	0	0	1	0	2.431	1054	0.599
M0016	1	0	0	0	0	0	2.315	1107	0.630
M0024	1	0	0	0	1	0	2.595	987	0.561
M0025	1	0	0	1	1	0	3.595	713	0.405
M0036	0	0	1	0	0	1	3.151	813	0.462
M0037	0	0	1	1	0	1	4.151	617	0.351
M0044	0	0	1	0	1	1	3.431	747	0.425
M0052	1	0	1	0	0	1	3.315	773	0.439
M0060	1	0	1	0	1	1	3.595	713	0.405
M0061	1	0	1	1	1	1	4.595	558	0.317
M0100	0	1	1	0	0	1	3.220	796	0.453
M0108	0	1	1	0	1	1	3.500	732	0.416

Résultats obtenus

Conditions expérimentales

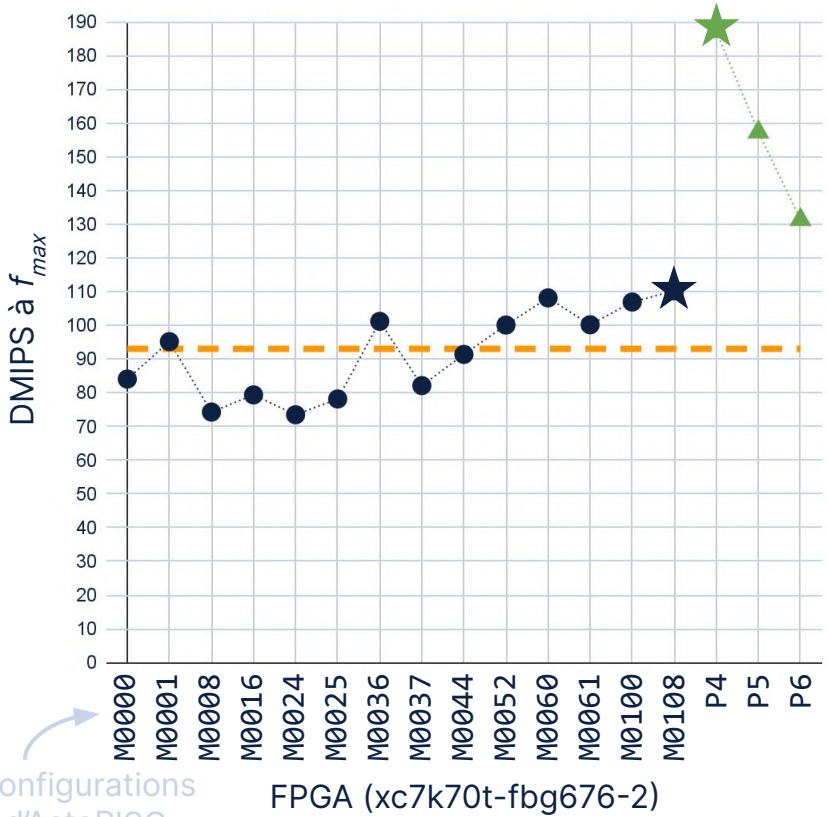
SoC pour AsteRISC

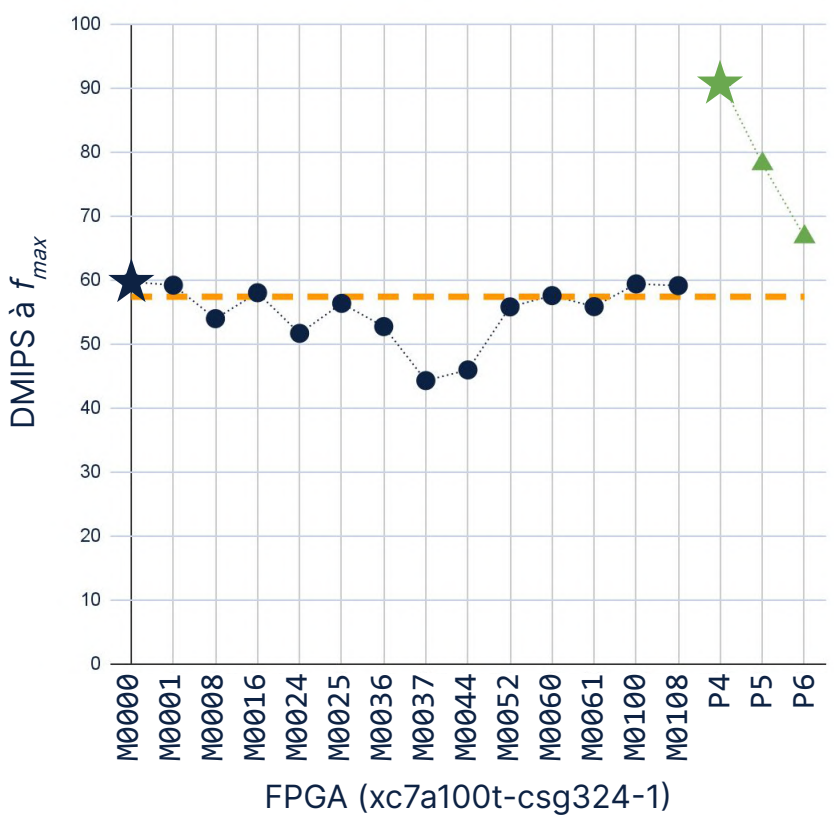


Résultats obtenus

Performances sur FPGA

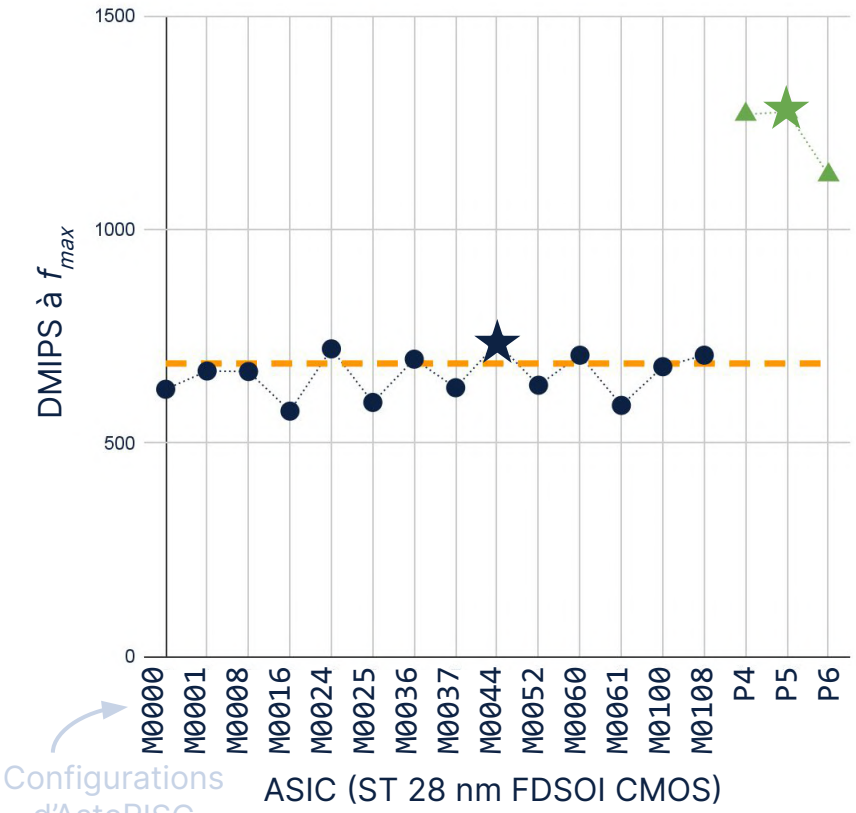
- AsteRISC non pipeline
- AsteRISC pipeline
- PicoRV32
- Meilleure performance

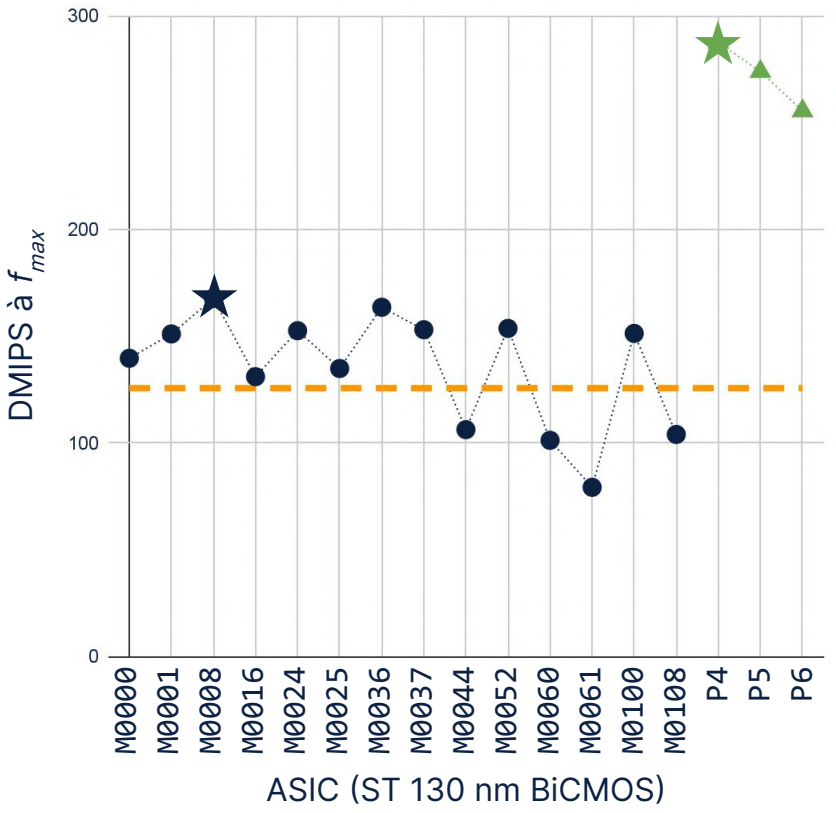




Résultats obtenus

Performances sur ASIC

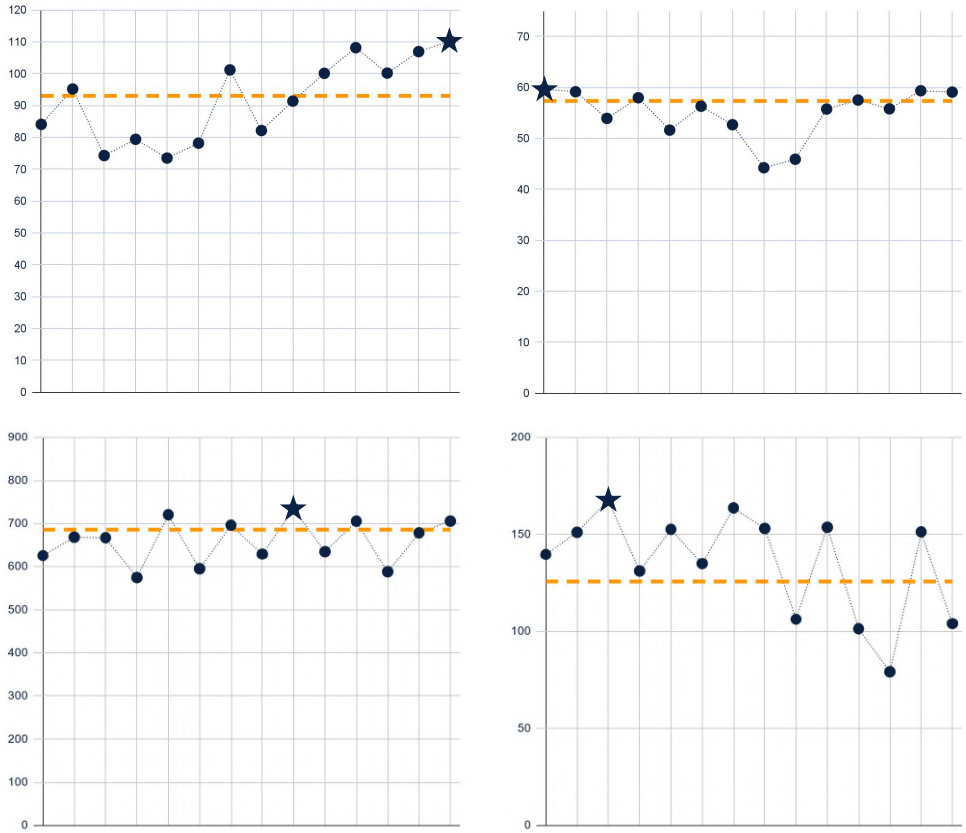




AsteRISC non pipeline
AsteRISC pipeline
PicoRV32
Meilleure performance

Résultats obtenus

Impact du choix de la cible technologique

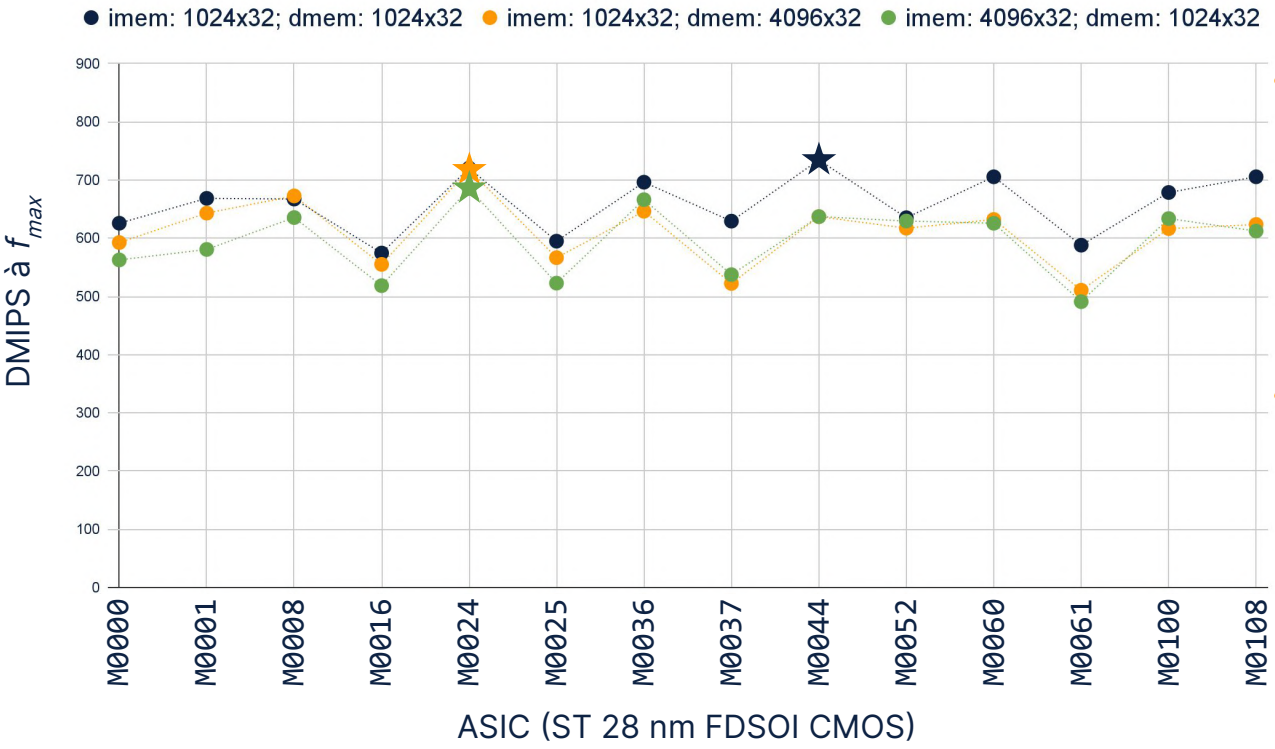


- Les allures sont très différentes
  - ➜ La meilleure configuration est différente dans les 4 cas !
  - ➜ Si une architecture donne de bonnes performances sur une technologie, ce n'est pas forcément le cas sur une autre
- Il y a toujours des configurations qui donnent de meilleurs résultats que le PicoRV32

Résultats obtenus

Impact de la taille des mémoires

- AsteRISC non pipeline
- ★ Meilleure performance



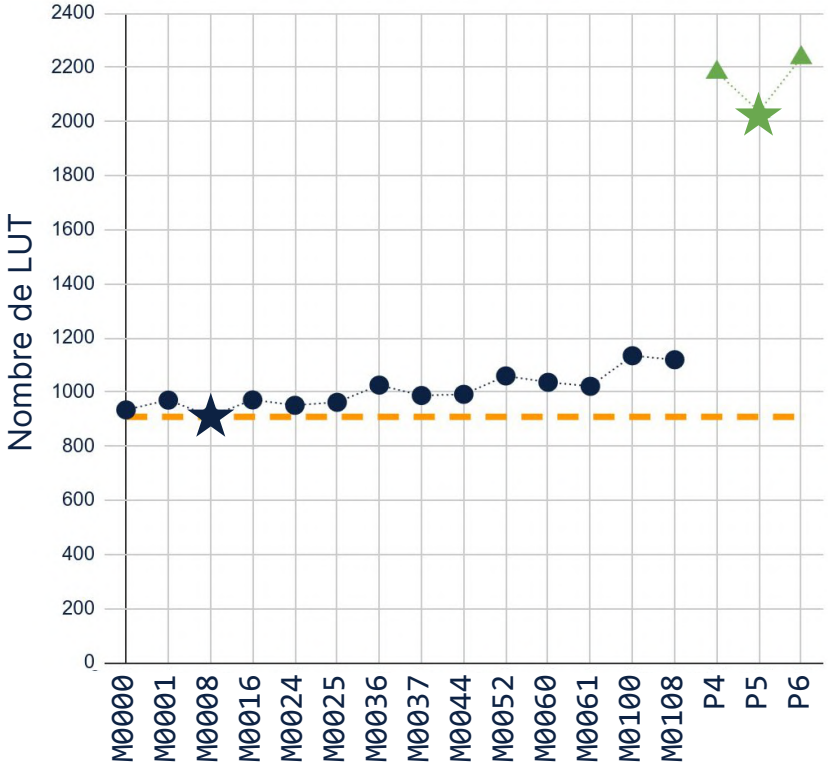
➜ Les configurations réagissent différemment au changement des caractéristiques des mémoires

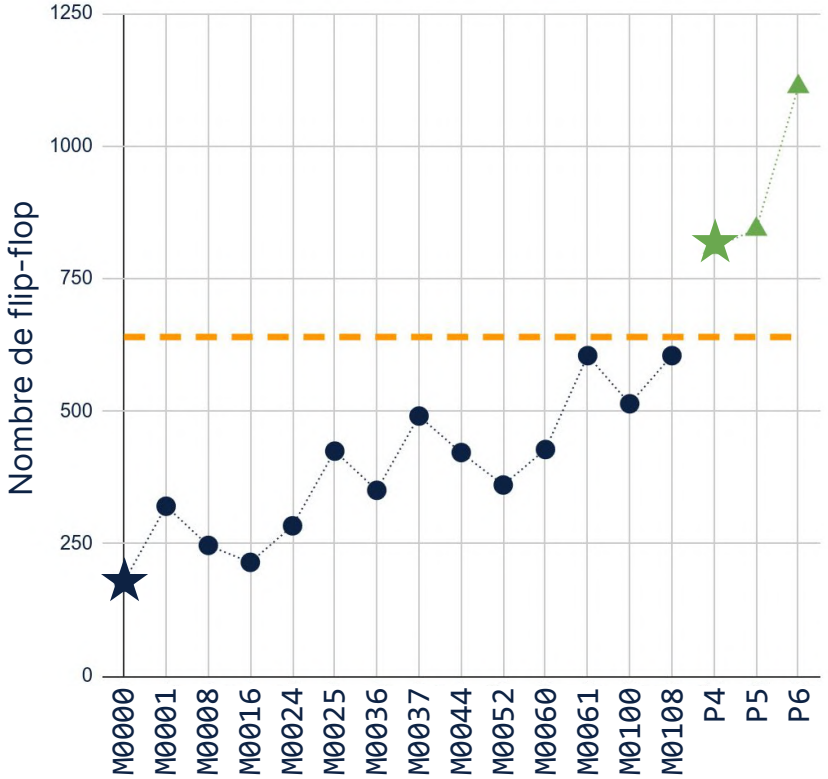
➜ Ce n'est parce qu'une configuration donne les meilleures performances dans un contexte que c'est le cas dans tous les contextes

Résultats obtenus

Utilisation des ressources sur FPGA

AsteRISC non pipeline
AsteRISC pipeline
PicoRV32
Le moins de ressources



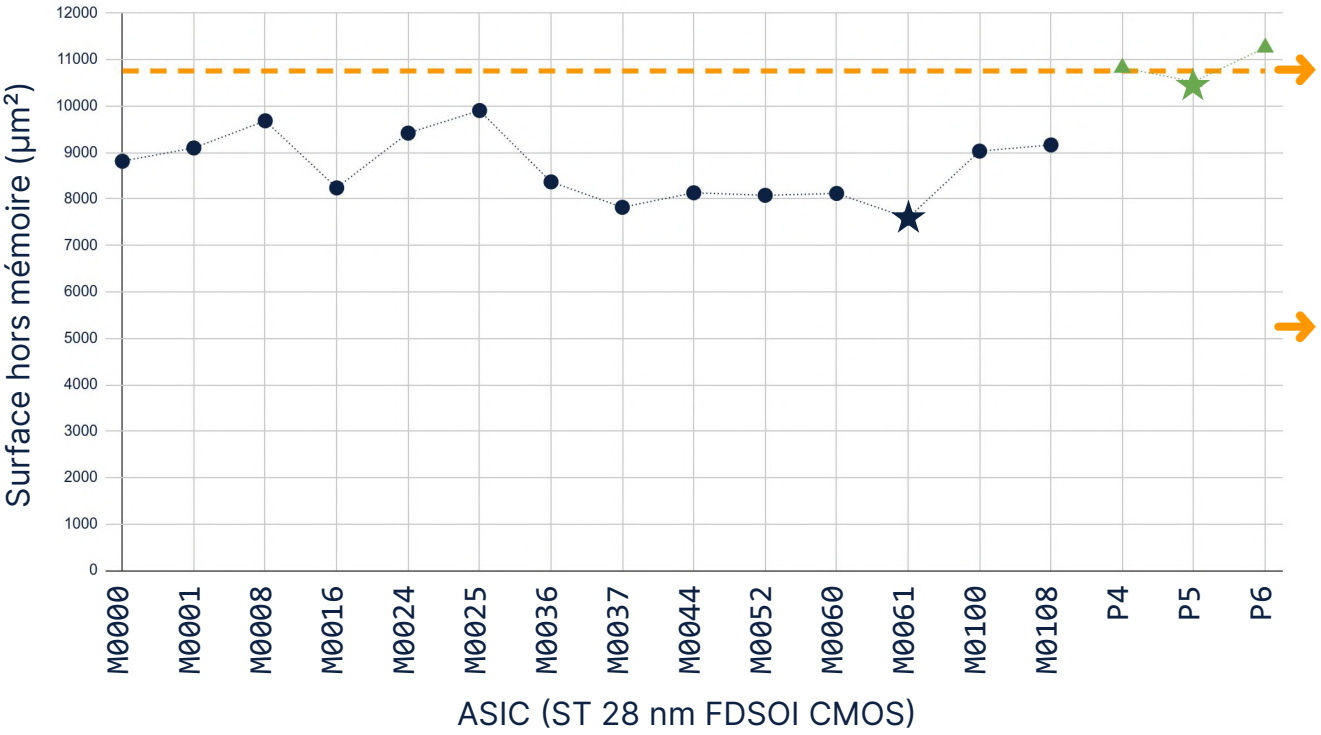


FPGA (xc7k70t-fbg676-2)

Résultats obtenus

Utilisation des ressources sur ASIC

- AsteRISC non pipeline
- AsteRISC pipeline
- PicoRV32
- Le moins de surface



Les configurations non-pipeline occupent toutes moins de surface que PicoRV32

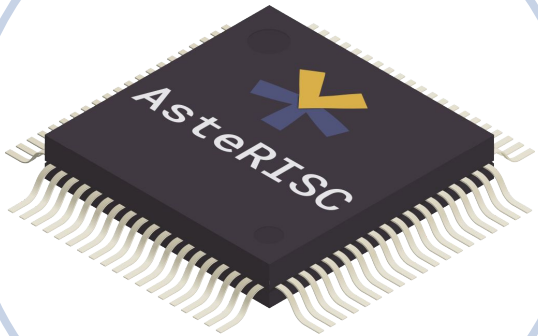
La configuration la moins surfacique ici est une des pires sur FPGA (M0061)

Conclusion et perspectives

Démonstration de l'intérêt des architectures flexibles

Moyen de chercher le meilleur compromis entre performances, ressources, et consommation

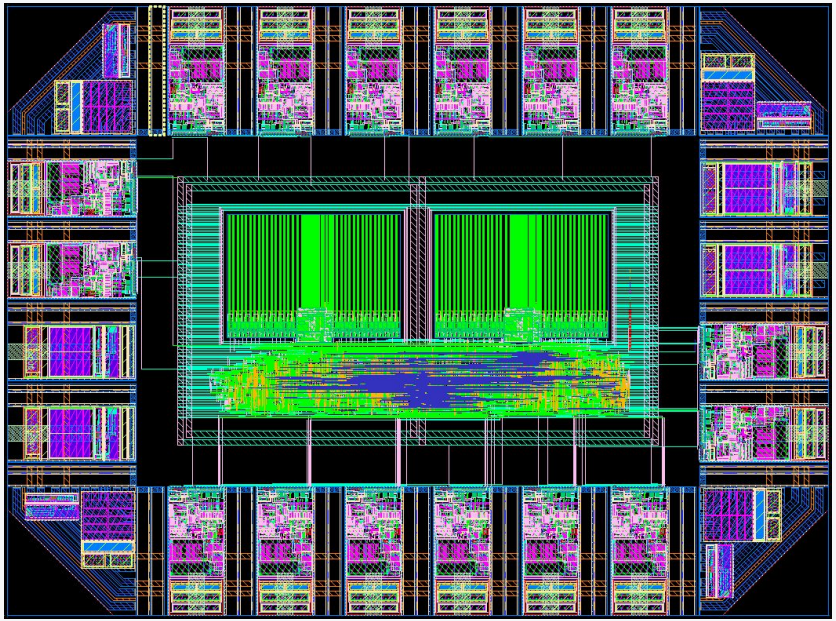
Possibilité d'obtenir un CPU optimisé, même dans les contextes de fortes contraintes



Augmentation de l'espace de conception qui peut être exploré avec AsteRISC

Développement de versions hybrides pipeline/non-pipeline

Comparaison avec la flexibilité à base de HLS/Chisel



AsteRISC

Un cœur RISC-V flexible

Jonathan Saussereau
Christophe Jégo
Camille Leroux
Jean-Baptiste Bégueret

jonathan.saussereau@ims-bordeaux.fr