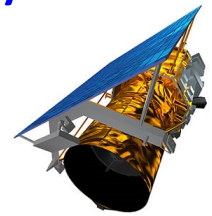
Introduction au Temps réel

Emmanuel Grolleau

isae Institut Supérieur de l'Aéronautique et de l'Espace
ENSMA
ENSIP Poitiers
Université de Poitiers

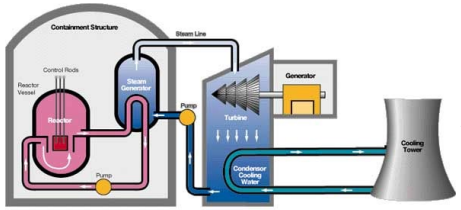
Définitions

- Un **procédé** est un système contrôlé par un système de **contrôle/commande**
  - Les moyens de calcul peuvent être embarqués



  - Ou non embarqués

- Le procédé contrôlé est plus ou moins **critique**



- De nombreux systèmes de contrôle/commande critiques sont soumis à des **contraintes de temps** inhérentes à la dynamique du procédé et de son environnement, ils sont **temps réel**

Fait-on de la validation pour le plaisir ?

1. LE TEMPS RÉEL FAIT PARTIE DE LA SÛRETÉ DE FONCTIONNEMENT DES SYSTÈMES CRITIQUES

Un besoin industriel imposé par des standards

- Normes sûreté de fonctionnement (*safety*)

Domaine	Niveaux de criticité					
Aviation civile (**DO178**/254)	DAL-E	DAL-D 10^{-3}/h	DAL-C 10^{-5}/h	DAL-B 10^{-7}/h	DAL-A 10^{-9}/h	10^{9}h>114000 ans
Automobile (**ISO 26262**)	Gestion de qualité	ASIL-A	ASIL-B/C	ASIL-D		
General (IEC 61508)	-	SIL-1	SIL-2	SIL-3	SIL-4	
Rail (CENELEC 50126/128/129)	-	SIL-1	SIL-2	SIL-3	SIL-4	
Impact sur la sûreté	Pas d'impact	Mineur	Majeur	Dangereux	Catastrophique	

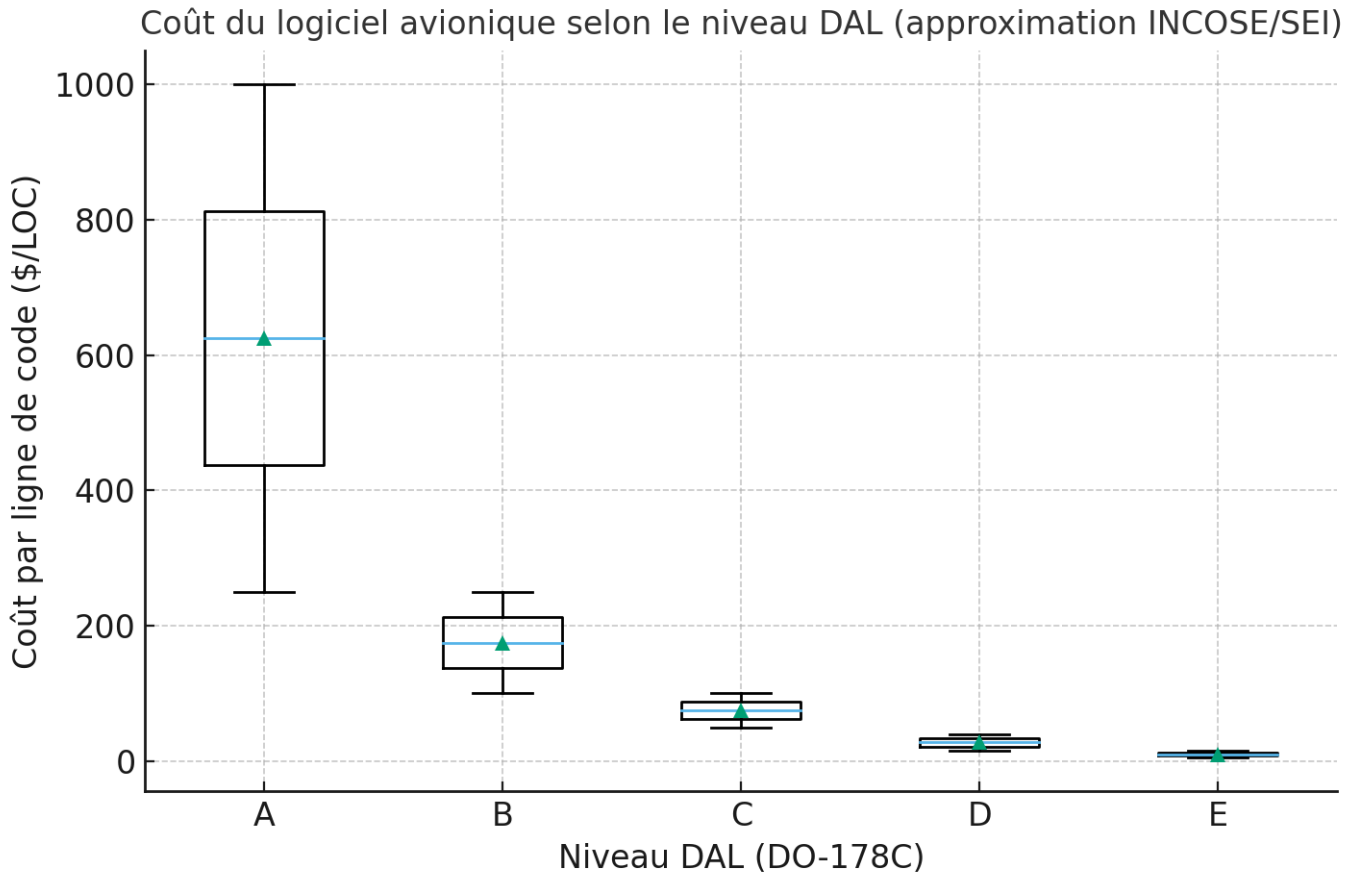
- Criticité mixte ≠ *Mixed Criticality* en ordonnancement temps réel

DO-178C

❑71 objectifs répartis dans 10 étapes

Niveau de criticité	Nombre d'objectifs à remplir	Remarques
DAL-E	0	
DAL-D	26	Intégrité du partitionnement, exigences de haut niveau consistantes, traçables jusqu'aux exigences système, code exécutable conforme aux exigences de haut niveau, exécutable sur la cible, test de couverture des exigences de haut niveau, gestion des configurations, etc.
DAL-C	62	Cycle de vie logiciel, exigences tracées, dérivées, démontrées, algorithmes corrects, archi logicielle consistante (mais pas forcément vérifiable), conforme à des standards, etc.
DAL-B	69	Vérifications concernant exécution sur cible, archi logicielle vérifiable, couverture de code plus exigeante
DAL-A	71	Couverture de code plus exigeante, y compris pour le code mort

Coûts logiciels en avionique civile



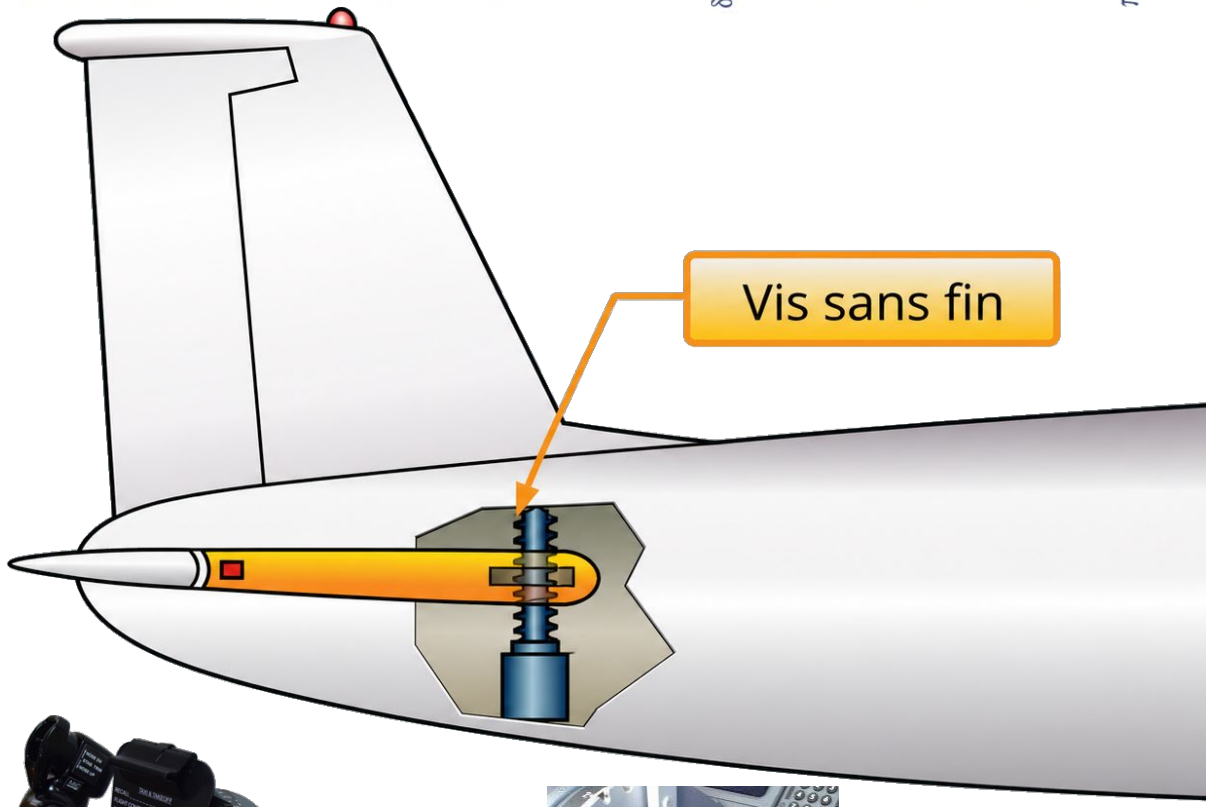
Deux exemples montrant l'importance de la sûreté de fonctionnement

- Le système MCAS du BOEING 737 MAX
- Accélération intempestive sur les voitures

Historique simplifié du Boeing 737 MAX

- 2010 : Annonce par Airbus de la sortie de l'A320 néo en 2016
  - Remplace les moteurs CFM56 par des moteurs LEAP
    - ✓ Consommation carburant annoncée -16%
    - ✓ Production CO2 -16%,
    - ✓ Production Nox -50%,
    - ✓ Bruit -15 dB
- 2011 : Annonce de Boeing de la sortie du 737 MAX pour 2017
  - Promesse : les pilotes de 737 ne nécessitent pas de formation spécifique
- Problème
  - Le LEAP est plus large que le CFM56
  - Génère une portance à haute incidence
  - Nécessite un train d'atterrissage réhaussé sur les Boeing 737
  - La position du PHR n'est pas la même en vol de croisière (position usuelle fait perdre de l'altitude)
- Solution Boeing 737 MAX
  - Utiliser le système automatique MCAS (Maneuvering Characteristics Augmentation System) lors des décollages et des vols de croisière

Commande de l'axe de tangage : Plan horizontal réglable (PHR, STAB en anglais)



- Plan qui possède la gouverne de profondeur en prolongement
- Plus lent au mouvement que la gouverne de profondeur
- En vol de croisière, sert de réglage de compensation en tangage de l'avion
  - Position dépend de l'altitude et de la vitesse
  - Permet de garder la gouverne de profondeur au neutre
  - Réduit la trainée p.r. gouverne de profondeur

Angle d'incidence

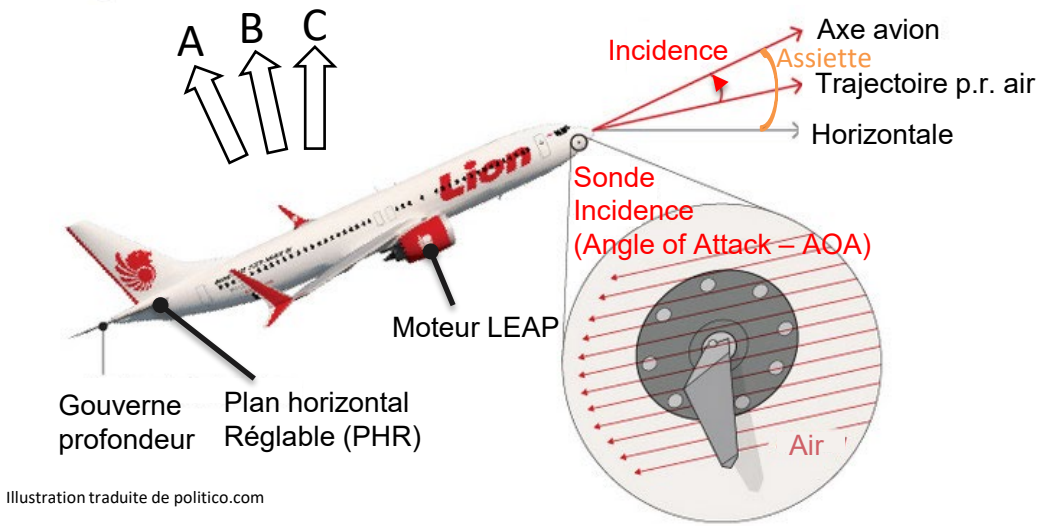
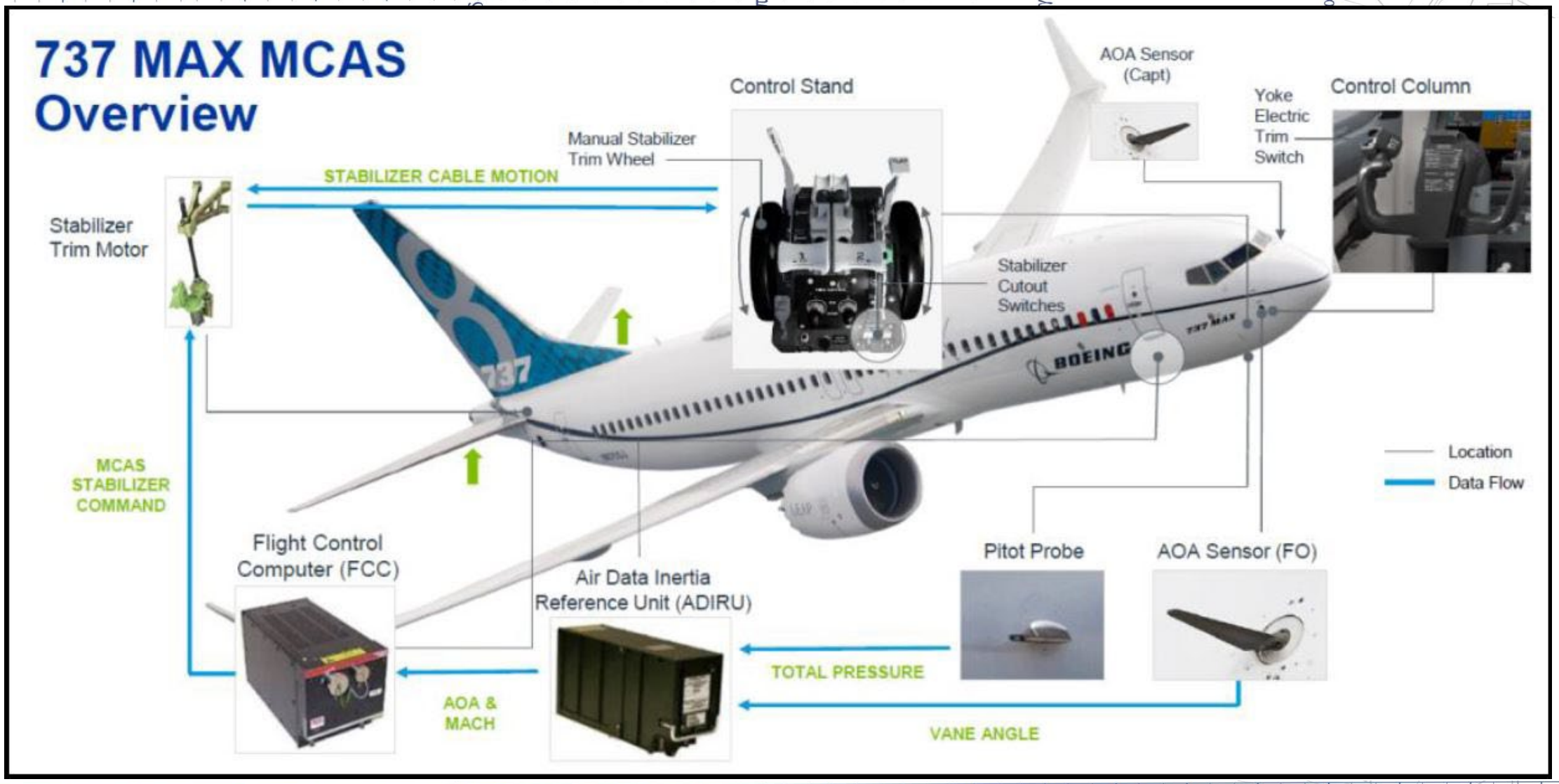


Illustration traduite de politico.com

- Avec une incidence élevée et assez de vitesse, l'avion prend de l'altitude
  - Pendant un décollage assiette 10° à 15°
  - Incidence 6° à 10°
- Car la poussée est dans l'axe avion
- La portance des ailes est
  - A - perpendiculaire à l'axe avion ?
  - B - perpendiculaire à la trajectoire ?
  - C - verticale ?

La portance des ailes est :

737 MAX MCAS Overview

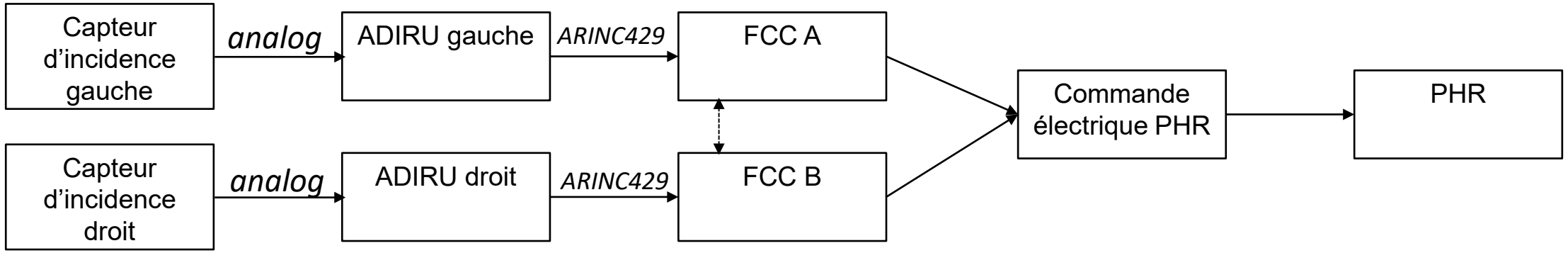


Principes de la tolérance aux fautes

- Le 0 défaut n'existe pas
- Chaque élément possède un taux de défaillance exprimé en défaillance par heure de fonctionnement λ.
  - Exemple: supposons qu'un calculateur et un capteur aient $\lambda=10^{-4}$.
  - Cela signifie qu'en moyenne, sur une grande population de calculateurs, on observe environ une défaillance pour 10'000 heures cumulées de fonctionnement
  - 10'000 heures = 1,14 ans – c'est beaucoup ?
  - Aujourd'hui avec plus de 2000 737 MAX en exploitation, on peut estimer qu'ils cumulent ~20'000 heures de vol par jour
  - On peut donc attendre une moyenne de deux défaillances par jour sur un élément avec $\lambda=10^{-4}$

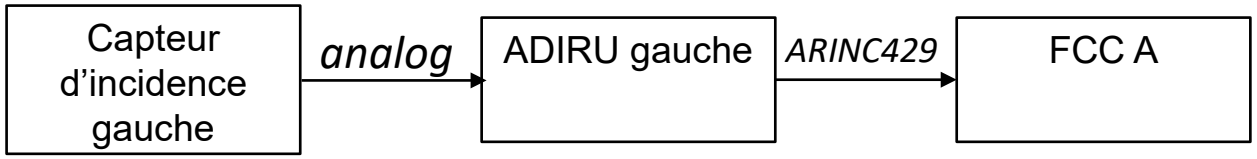
Principe du MCAS au décollage avant 2020

- lors de la phase de décollage
  - Tant que incidence > 17°
    - ✓ Monter la vis sans fin du PHR pendant 10 secondes
    - ✓ Attendre 5 secondes
- Source d'angle d'incidence
  - 1 calculateur ADIRU, utilisant 1 sonde d'incidence, celle du côté du Flight Control Computer (FCC) actif



- En cas de différences entre les 2 sondes d'incidence (communication entre les deux FCC)
  - Si option achetée, affichage d'un message « AOA disagree »
  - Mais le MCAS fonctionne tout de même si sa sonde indique incidence > 17°

Une seule source décide



- Supposons un taux de défaillance de $\lambda=10^{-4}$ sur chaque élément de la chaîne, et $\lambda=10^{-3}$ sur les capteurs d'incidence
- En 2019 avec ~350 B737 MAX en service, en supposant 0,2 heures de vol en phase de décollage par jour par avion
- 70 heures de « phase de décollage » par jour pour environ 1 défaillance « critique au décollage » pour 1000 heures.
- On peut s'attendre à 1 défaillance critique toutes les 2 semaines

Une première version du MCAS problématique

- Lion Air 610, crash le 29/10/2018, 189 personnes à bord
- Ethiopian Airlines 302, crash le 10/03/2019, 157 personnes à bord
- Vraisemblablement des dizaines d'incidents MCAS non documentés, n'ayant pas mené au crash
  - Mon $\lambda=10^{-3}$ sur les sondes d'incidence était optimiste car ne prenait pas en considération les erreurs maintenance
- Coût direct pour Boeing > 21 Mrd$, coût indirect dû aux pertes de ventes 67 Mrd$, sans parler de l'image ébranlée du constructeur

Leçons apprises

- Problèmes liés à la pression commerciale
  - « Faire passer » un 737 MAX pour un 737 NG
  - Pas de re-certification pilotes certifiés auparavant sur 737 NG
- Problème d'absence de prise en compte de **redondance**
  - Il y a redondance, mais en *hot standby*, la redondance est primordiale dans le sûreté de fonctionnement, mais on voit qu'il existe plusieurs types de redondance
- Détection difficile du problème par **arbre de fautes**
  - Pilotes non conscients du fonctionnement MCAS
  - Difficulté pour le pilote d'avoir une **autorité** suffisante pour contrer l'action du MCAS sur le PHR
  - Répétition d'actions automatiques de corrections du PHR
- Nécessité de prise en compte au **niveau systémique**
  - Y compris humain
  - Modélisation de l'autorité
- Nécessité de faciliter les retours de lanceurs d'alerte liées à la sûreté

Criticité du MCAS

- Pré-2020, hors des phases de décollage, le MCAS était de criticité DAL-C
- Sachant que lors des phases de décollage, le pilote était supposé pouvoir reprendre la main sur le PHR, quel est le niveau de criticité pré et post-2020 de la fonction MCAS au décollage ?

Sachant que lors des phases de décollage, le pilote était supposé pouvoir reprendre la main sur le PHR, quel est le niveau de criticité pré et post-2020 de la fonction MCAS au décollage ?

Cas d'éccélérations intempestives

- En 2009 plusieurs accidents graves aux USA

  - Notamment Toyota Lexus
  - Très médiatisé et choquant l'opinion publique : accident famille de 4 personnes avec appel 911 durant la conduite
  - 09/2009 Rappel 3,8 M véhicules pour tapis de sol
  - 01/2010 Rappel 2,3 M véhicules pour « pédale collante »
  - 02/2010 Rappel étendu pour ajouter un *Brake Override System*
    - ✓ *Si frein et accélération simultanés, alors décélération*
    - ✓ *Module nécessitant un front montant sur le freinage*
- En 2010, la NHTSA (*National Highway Traffic Safety Administration*) avait reçu des plaints couvrant 93 morts et plus de 300 blessés depuis 2000 sur cette marque qui pourraient être liés aux accélérations intempestives
- Plusieurs procès civils et collective actions
  - *2014 Toyota paie une sanction financière de $1,2M*[d]

Qq soucis recensés par les experts des plaignants dans l'affaire Bookout v. Toyota sur Toyota Camry 2005

- documentation et spécifications jugées insuffisantes
- mécanisme de watchdog réinitialisé par une interruption périodique, ne permettant pas de détecter la mort de certaines tâches importantes
- utilisation de récursion avec pile sous-dimensionnée
- environ 10000 variables globales alors que plus de 70% d'entre elles auraient dû être statiques/locales
- partage de variables globales entre tâches sans mécanisme d'exclusion mutuelle
- variables partagées non déclarées volatiles
- alors que ce système est rétrospectivement ASIL-C, 80000 violations de la norme MISRA C (proposée initialement en 1998 qui sera recommandée dans l'ISO 26262 à partir de 2011)
- architecture de tolérance aux fautes jugée insuffisante et présence de défaillances ponctuelles
- redondance non ségréguée sur la lecture de position de la pédale d'accélérateur
- 67 fonctions présentant une complexité cyclomatique supérieure à 50
- complexité cyclomatique d'une des fonctions – 146 (au delà de 50, fonction réputée pas testable, au-delà de 75, corriger un bug insère généralement un autre bug),pas de gestion de configuration
- processus de revue, de suivi des anomalies et de gestion des modifications jugés insuffisamment documentés, etc.
- *Ces conclusions proviennent des experts des plaignantes dans l'affaire Bookout v. Toyota. Toyota les a contestées. L'étude NHTSA–NASA de 2011 n'avait pas identifié de défaut électronique expliquant les accélérations intempestives examinées, sans toutefois démontrer l'absence de toute défaillance logicielle possible.*

Leçons apprises

- A été l'un des éléments ayant poussé l'adoption par l'industrie de l'ISO 26262 en 2011
  - (en comparaison DO-178 sortie en 1982)
  - Reprend les grands principes de la sûreté de fonctionnement tels que décrits dans la DO-178, de façon plutôt descriptive
  - Parallèlement à AUTOSAR qui propose process et architecture permettant de suivre (de façon prescriptive) l'ISO 26262
  - Introduit la notion d'ASIL définie en fonction de Sévérité, Exposition et Contrôlabilité
  - Demande à suivre des règles de programmation (ex: MISRA-C)
  - Demande à évaluer la complexité logicielle (ex: cyclomatique) du code

L'automobile est sur le point d'opérer de grands virages

❑Classification SAE J3016

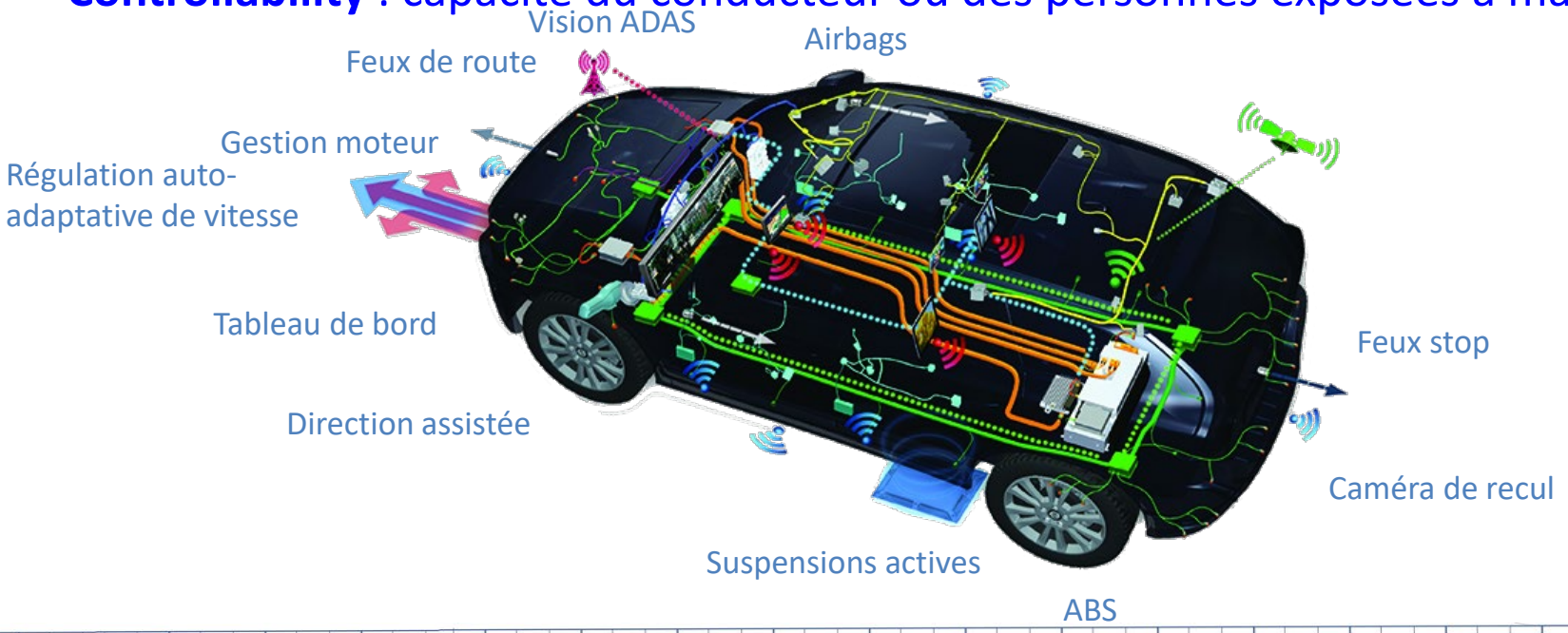
Niveau	Qui conduit et surveille ?	Situation actuelle
0	conducteur, avec alertes ponctuelles	courant
1	conducteur, assistance longitudinale **ou** latérale	courant
2	conducteur, assistance longitudinale **et** latérale	très répandu
3	système ; conducteur disponible pour reprendre	premiers déploiements très limités
4	système, sans reprise humaine dans un domaine défini	robotaxis géorestreints
5	système partout et en toutes circonstances	inexistant commercialement

Ce que nous appellerons ADAS (Advanced Driver Assistance Systems) dans la suite

Quelques fonctions automobile

Automobile (ISO 26262)	Gestion de qualité (QM)	ASIL-A intégrité faible	ASIL-B intégrité modérée	ASIL-C intégrité élevée	ASIL-D intégrité maximale

- L'ASIL est déterminé à partir de trois paramètres :
- **S — Severity** : gravité des blessures
- **E — Exposure** : fréquence d'exposition à la situation dangereuse
- **C — Controllability** : capacité du conducteur ou des personnes exposées à maîtriser la situation.



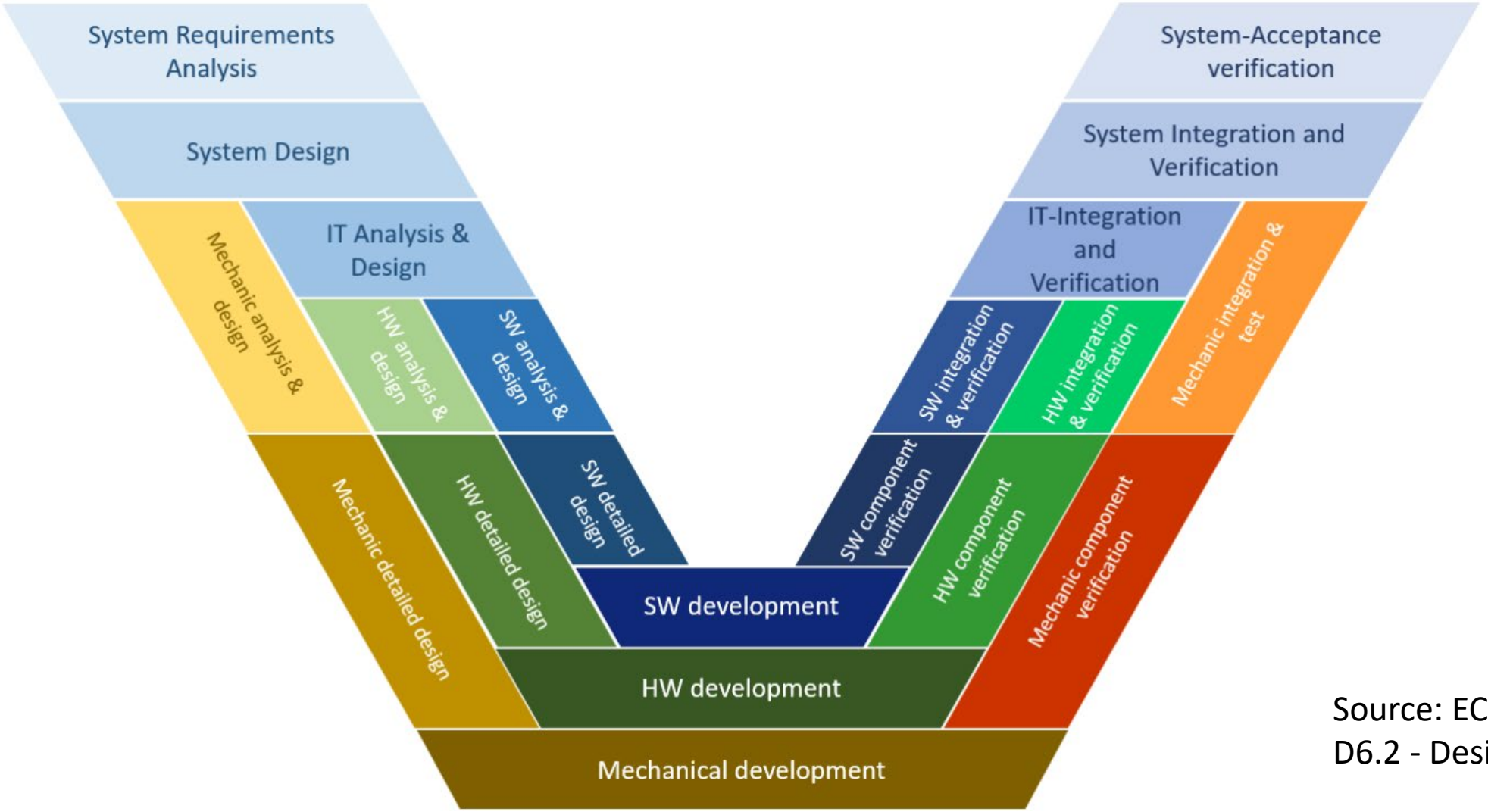
Pourquoi la régulation auto-adaptative de vitesse en aide à la conduite (ADAS) n'est que B ou C et pas D ?

Quelle est la criticité du freinage ABS (Antiblockiersystem)

Quelle est la criticité des airbags ?

Quelle est la criticité de la partie vision ADAS (Advanced Driver Assistance Systems) ?

Cycle en V est adapté aux cas où les exigences critiques changent peu



Source: ECSEL H2020 Comp4Drones D6.2 - Design Tools

Où se trouve le temps réel d'ETR ?

- Respect des échéances : exigences non fonctionnelles
- Exigences liées aux performances
  - Contraintes de bout-en-bout
  - Pouvant se décliner en échéances sur les tâches et les messages
- Dans la DO-178C, en particulier dans la phase *Verification of Outputs of Software Design Process*
  - Traçabilité des exigences de haut niveau au bas niveau, y compris sur la cible (A4-1 à 6)
  - Algorithmes sont corrects (A4-7)
  - Architecture logicielle vérifiable et consistante avec les exigences y compris sur la cible (A4-8 à 11)
  - Architecture logicielle conforme à des standards (A4-12)
  - Intégrité du partitionnement (A4-13)

Positionnement validation temporelle

- Tard dans le cycle de vie
- Requiert WCET
  - Requiert souvent code généré sur cible
- Risque industriel important
  - Compensé par sur-dimensionnement
- Enjeu industriel
  - Permettre d'analyser un système temps réel plus tôt

Idées reçues sur le temps réel

- Les processeurs sont de plus en plus puissants, donc on n'aura plus besoin de vérifier le respect des contraintes de temps

- Le temps réel c'est plus rapide

- Le temps réel c'est compliqué

Difficultés/limitations des approches

- Nombre ∞ de scénarii possibles
- Nombre d'états très important, qui peuvent en plus requérir un temps compact
- Complexité NP-difficile dès qu'on sort de cas académique non réalistes

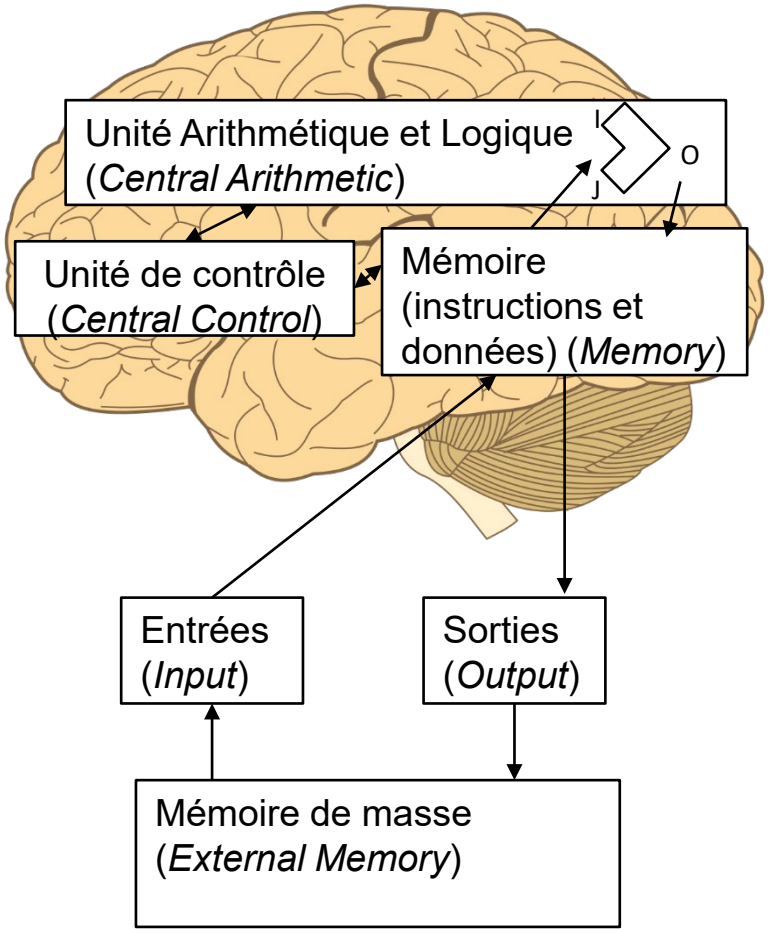
  - ✓ une tâche indépendante?
  - ✓ une préemption ou migration à coût nul?
  - ✓ une E/S non suspensive?
  - ✓ un seul mode de fonctionnement?
  - ✓ un système ne fonctionnant qu'en mode nominal?
  - ✓ un Worst-Case Execution Time sûr et pas pessimiste?
  - ✓ …

Depuis les 1940’s tout a changé, et rien n’a changé

2. DES PROCESSEURS OPTIMISÉS POUR LES PERFS MOYENNES

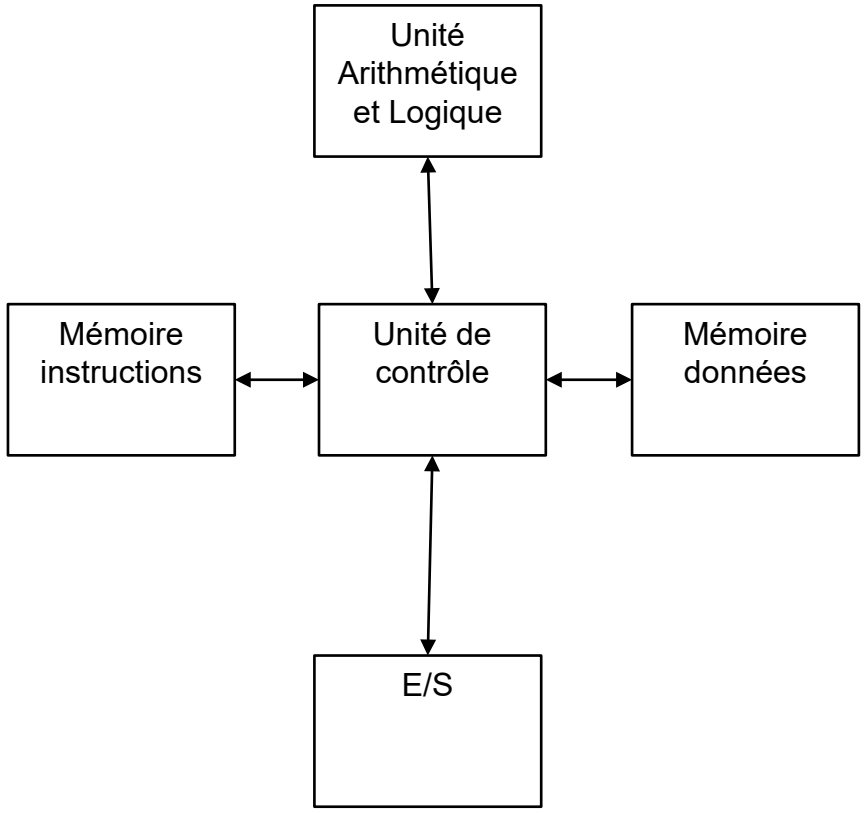
2.1 LE GOULOT D'ÉTRANGLEMENT DE LA MÉMOIRE

Architecture von Neumann

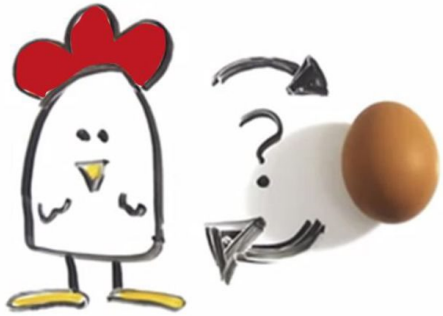


- Proposée en 1945 pour l'ENIAC
- Utilisation du binaire
- Agrégation de E-éléments (portes logiques)
- Utilisation d'une horloge commune
- Circuits séquentiels créant des registres

Architecture Harvard et Harvard modifiée

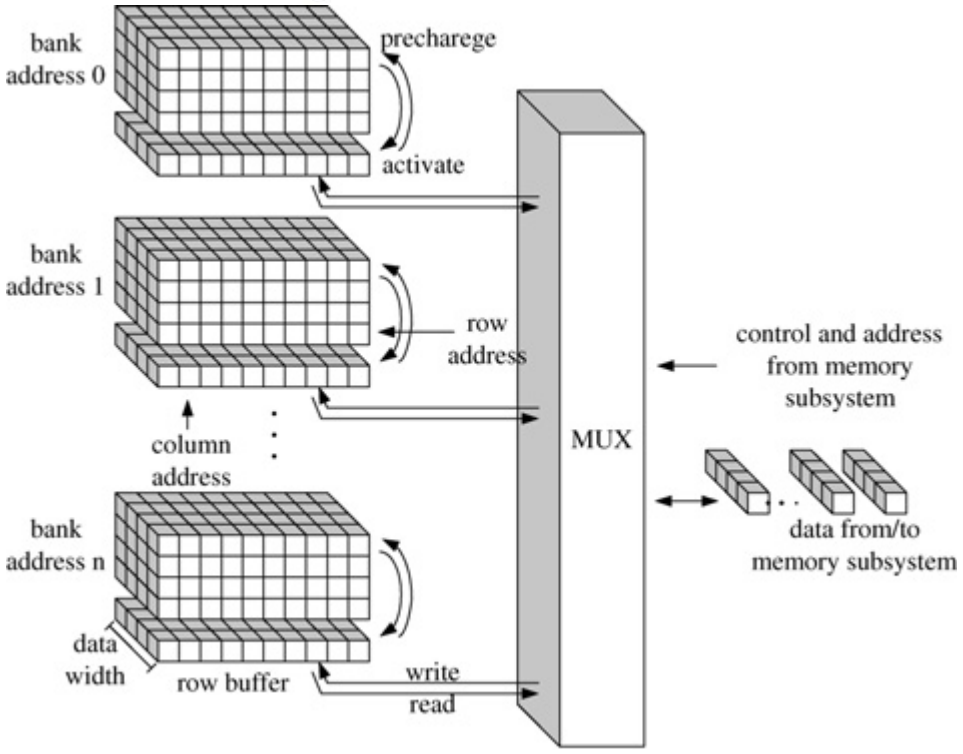


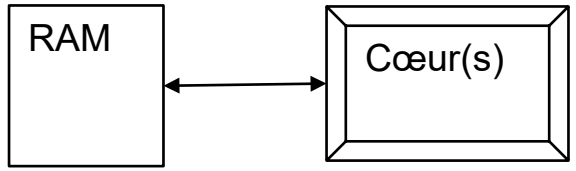
- D'après l'ASCC Mark I développé par IBM pour l'Univ Harvard en 1944
- On ne peut utiliser de données comme instructions, ni voir des instructions comme données
- Harvard modifié permet l'un ou l'autre, ou les deux



- Les ordinateurs modernes, von Neumann ou Harvard modifié ?

Structure interne d'une SDRAM





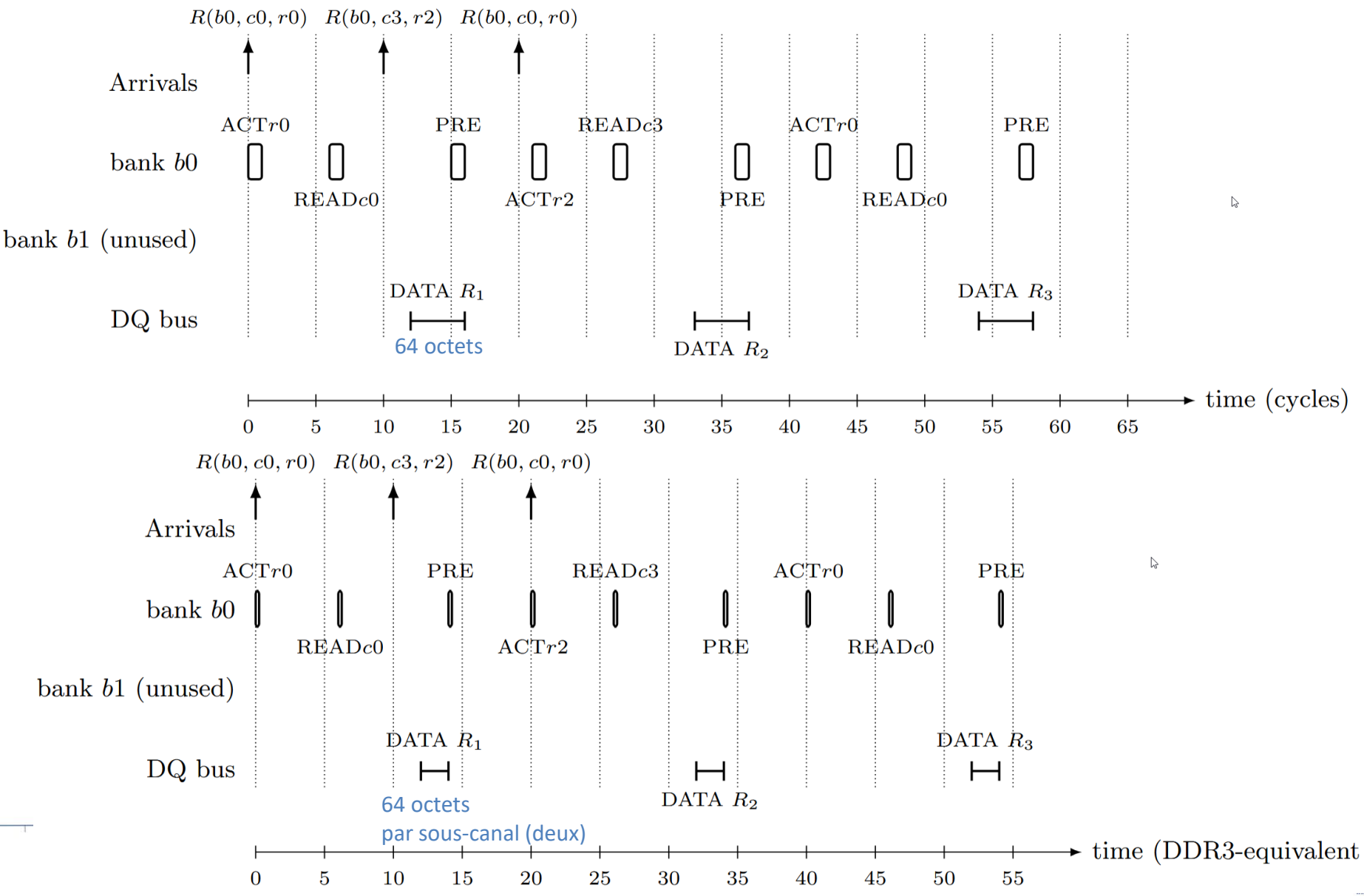
Source: W. Jang, D.Z. Pan, « An SDRAM-Aware Router for Networks-on-Chip »,
IEEE Tr. on Computer-Aided Design of Integrated Circuits, 29(10) 2010

- Avant de R/W une colonne, la ligne doit être activée par Activate(bank,row)
  - R/W(bank, column) – Column 64 octets (128 pour DDR5)
- Délais nécessaires pour chaque opération
- Possibilité de paralléliser (en pipeline) des opérations sur des banques différentes

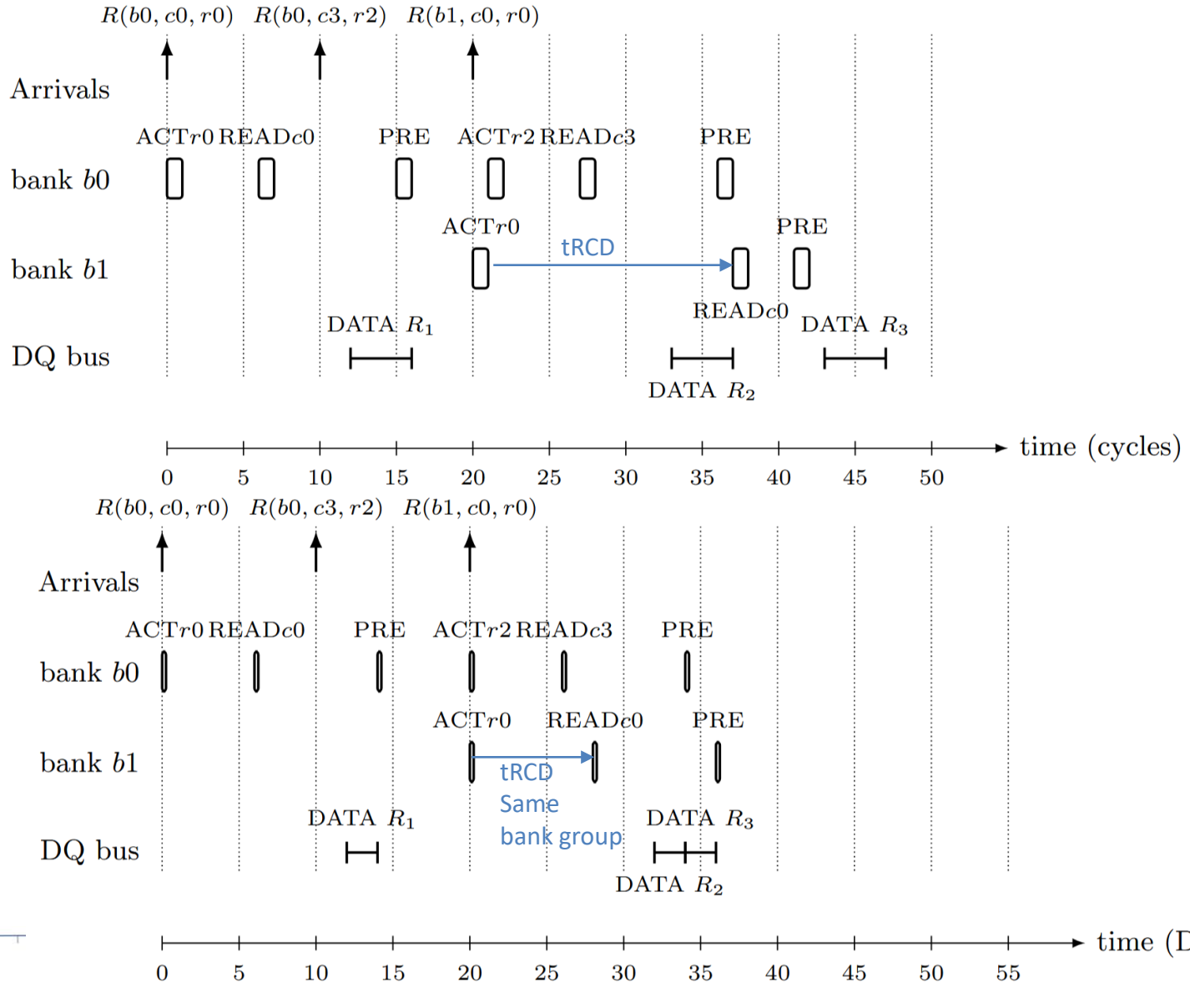
Quelques délais technologiques

	DDR1	DDR2	DDR3	DDR4	DDR5
Freq (MHz). Horloge réelle = ½ Freq -> 1 cycle DDR1-200MHz=10 ns	**200**	**400**	**800**	**1600**	**3200**
CL (Ligne -> Colonne, i.e. data)	2	3	6	11	24
tRCD (ACT->Read)	2	3	6	11	24
tRP (PRE->ACT)	2	3	6	11	24
tRAS (**ACT->PRE**)	5	8	15	28	48
tRC (Row cycle) (ACT ligne i->ACT ligne j) = tRAS+tRP	7	11	21	39	72
tRTP (**RD-> PRE**)	1	2	4	8	12
tWR (WR recovery)	2	3	6	12	20
tCCD (RD->RD)	1	2	4	4	8

DDR3 800 vs DDR4 3200



DDR3 800 vs DDR4 3200

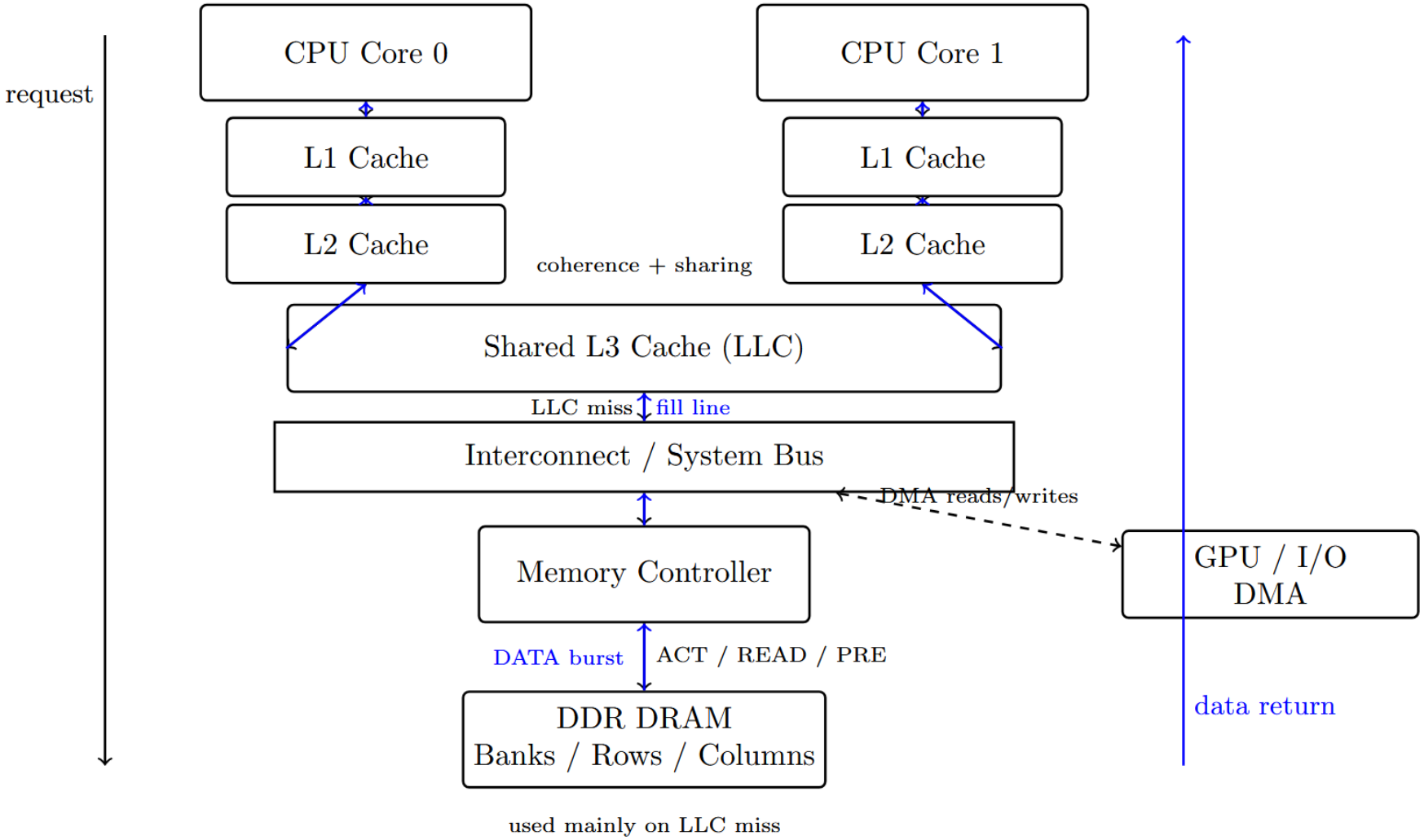


Il manque ici tRRD qui est un délai minimum entre deux ACT, dépendant des groupes de banques

Mémoire centrale /suite

- Aujourd'hui comme en 1945, la mémoire est un goulot d'étranglement
- Lire plusieurs mots de la même ligne à la suite est intéressant
  - Intérêt du prefetch
  - De l'alignement des adresses des variables avec les mots machine
  - La DDR est accédée par ligne de cache (1 ligne en DDR3, jusqu'à 2 en DDR5)
- Entrelacer les accès à différentes banques est intéressant
- Le temps d'accès à une variable est variable en fonction de l'ordre des accès, les requêtes à effectuer avant, l'ordonnancement interne des demandes d'accès par le contrôleur mémoire
  - Exemple : 34 cycles d'attente entre troisième requête et réponse de la DDR5 3200. Si processeur à 4 GHz, cela se traduit par 4*34/3,2>42 cycles du processeur! On ne compte même pas la contention d'accès au bus mémoire ni les requêtes des autres cœurs venant ralentir les requêtes du cœur qui nous intéresse.
  - Le cache mémoire joue un rôle très important de diminution des délais d'accès à la mémoire en faisant jouer le **principe de localité** et de **mémoire cache**.
- Il est quasiment impossible de prédire une durée d'exécution sur un processeur moderne qui ne soit pas très pessimiste p.r. cas moyen
  - Caches et TLB, succès au défaut de ligne ouverte, conflits de banques, réordonnancement par le contrôleur mémoire, rafraîchissement DRAM, contentions mémoires entre cœurs, cohérence des caches, spéculation et exécution hors ordre, etc.

Architecture générale



On remplace la mémoire par une génération offrant deux fois plus de bande passante. Le WCET d'une tâche diminue-t-il nécessairement ?

On exécute une tâche un million de fois. Le maximum observé est de 80 µs. Quelle affirmation est justifiée ?

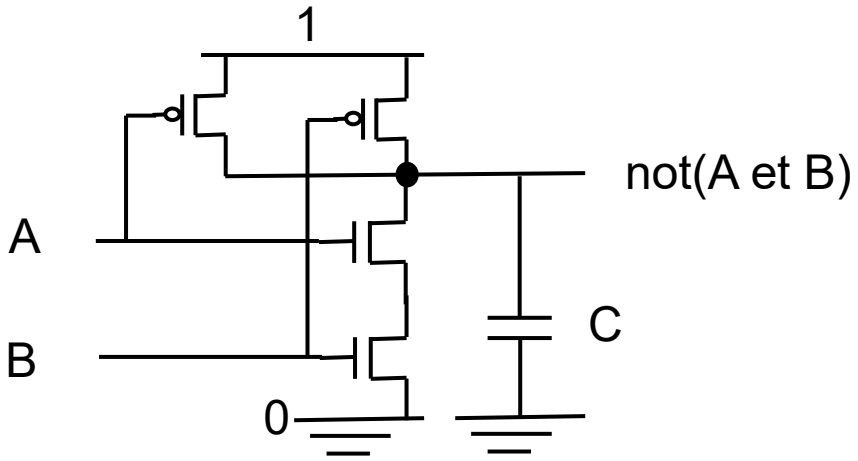
Une mémoire associative

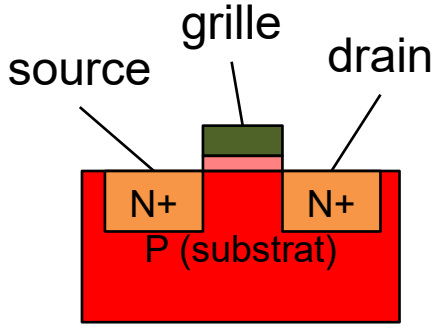
- Chaque ligne du cache contient un tag représentant l'adresse de la ligne en mémoire centrale, et le contenu
- A chaque passage de ligne en entrée ou sortie du cœur, chaque cache verra vérifier en une seule opération si l'adresse correspond à un tag présent en cache. Si oui:
  - Si lecture, c'est le cache qui répond à la requête
  - Si écriture, ligne remplacée dans le cache, en plus de la requête à remplacer dans la RAM qui peut être délayée sans problème
    - ✓Sauf cas multiprocesseur et donnée partagée, ou migration tâche l'utilisant
- Coût d'une migration d'un cœur à un autre

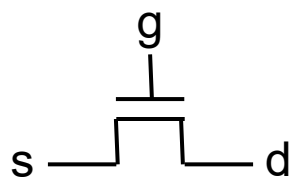
2.1 LA MONTÉE DU PARALLÉLISME

Transistors

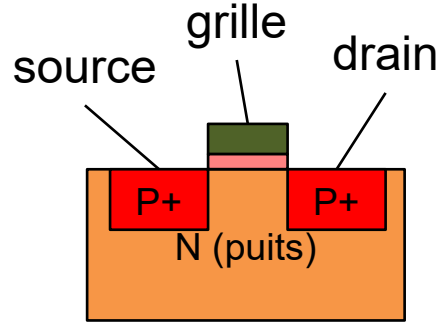
A	B	$\overline{AB}$
0	0	1
0	1	1
1	0	1
1	1	0

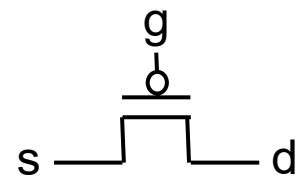






Passant si g=1

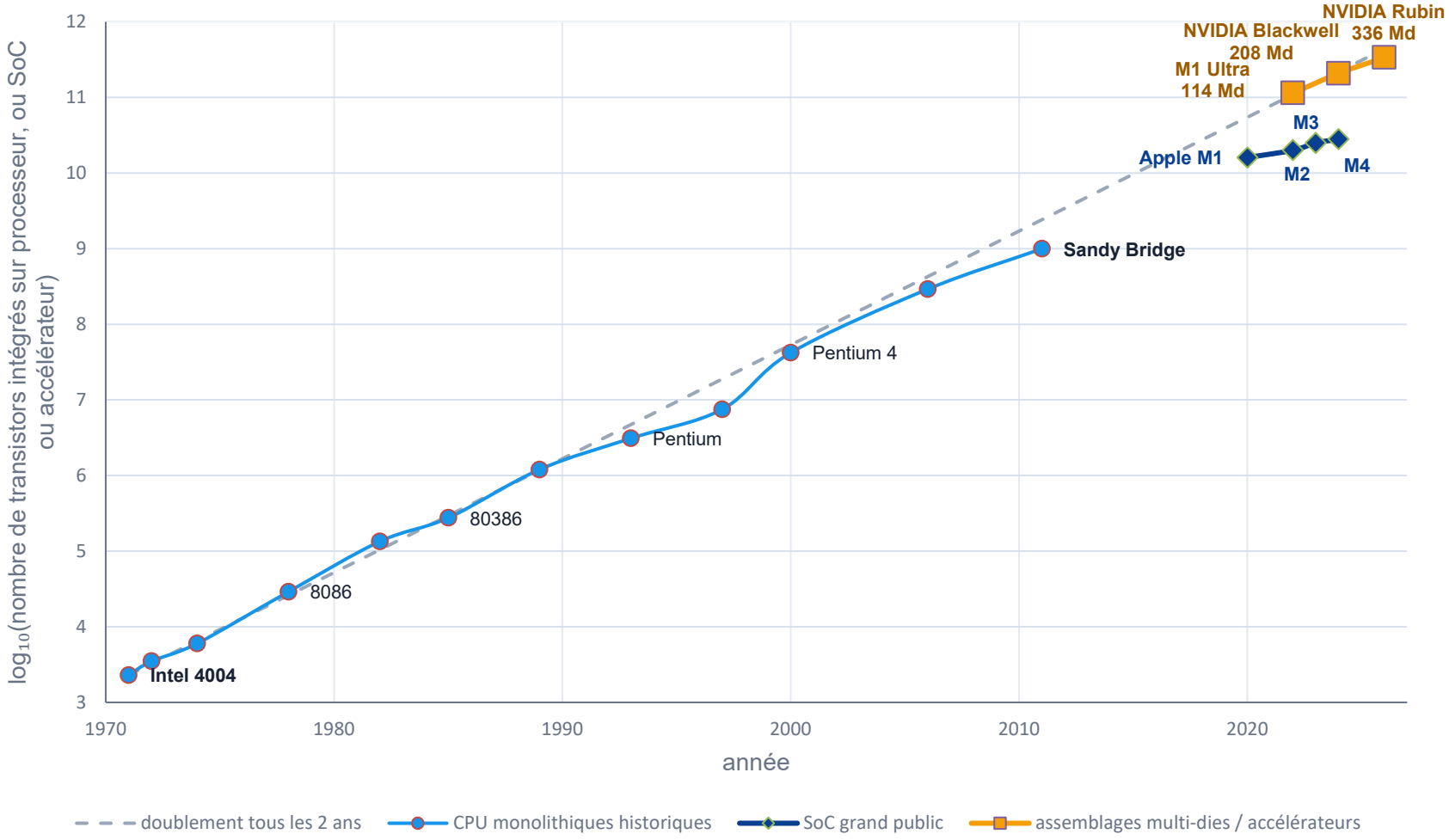




Passant si g=0

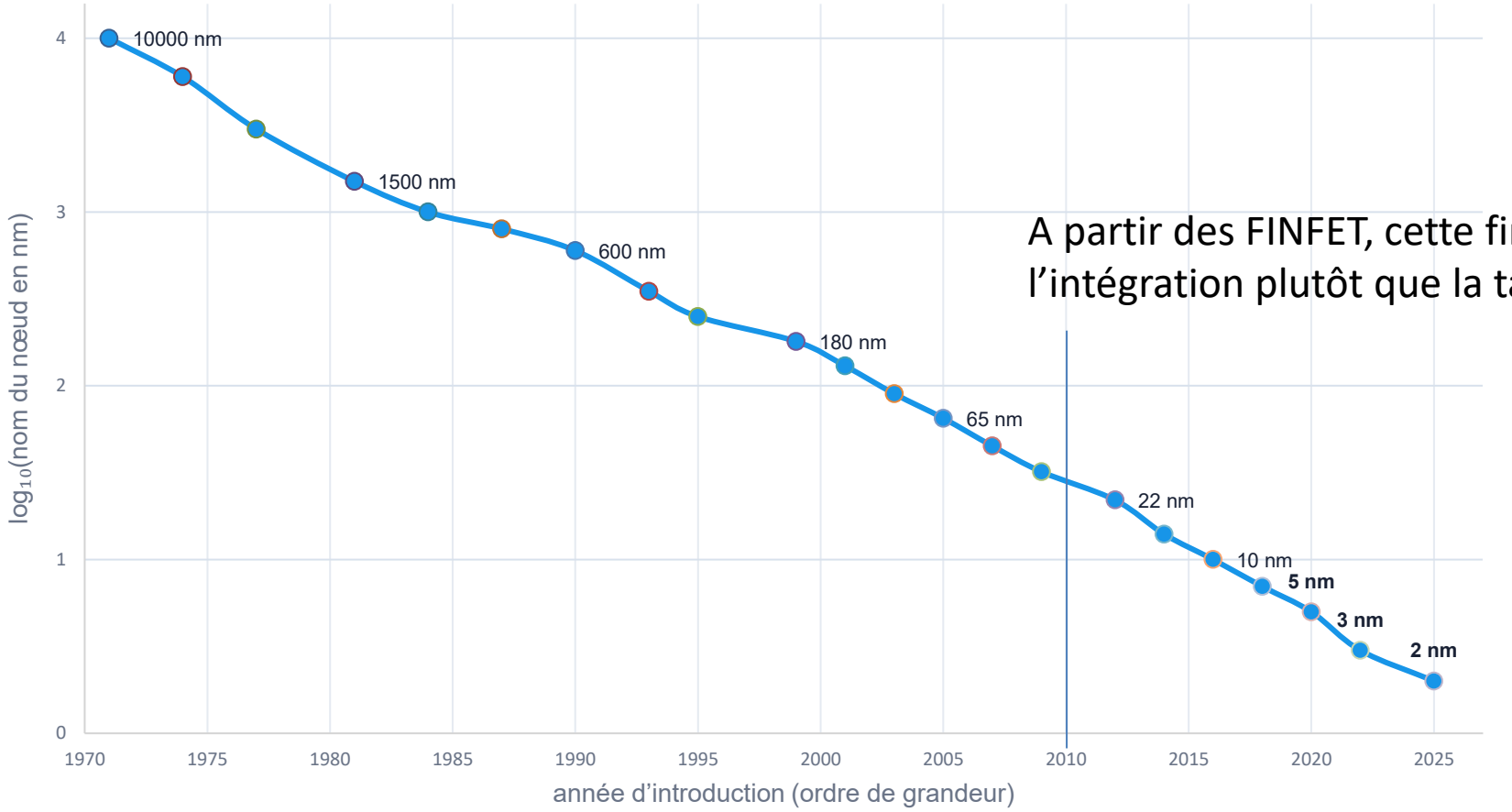
- Puissance consommée :
  - P = Pdyn + Pstatique
  - $P=\alpha \times CL \times Vdd^2 \times f + Vdd \times Ifuite$

Des promesses de la loi de Moore



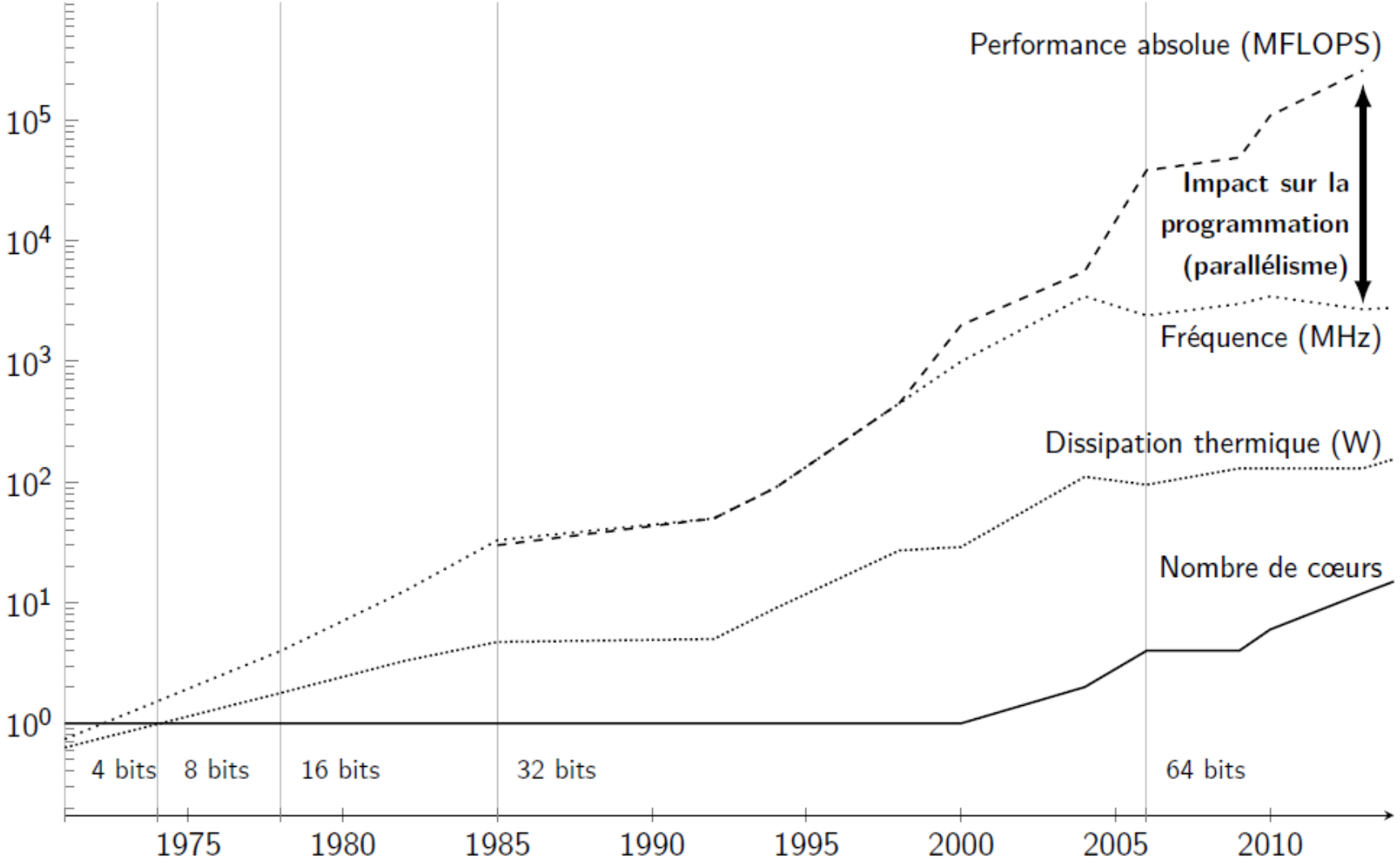
- Moore 1965: x2 tous les ans
- 1975: x2 tous les 2 ans

A la finesse « théorique » de gravure



A partir des FINFET, cette finesse « théorique » prend en compte l'intégration plutôt que la taille individuelle

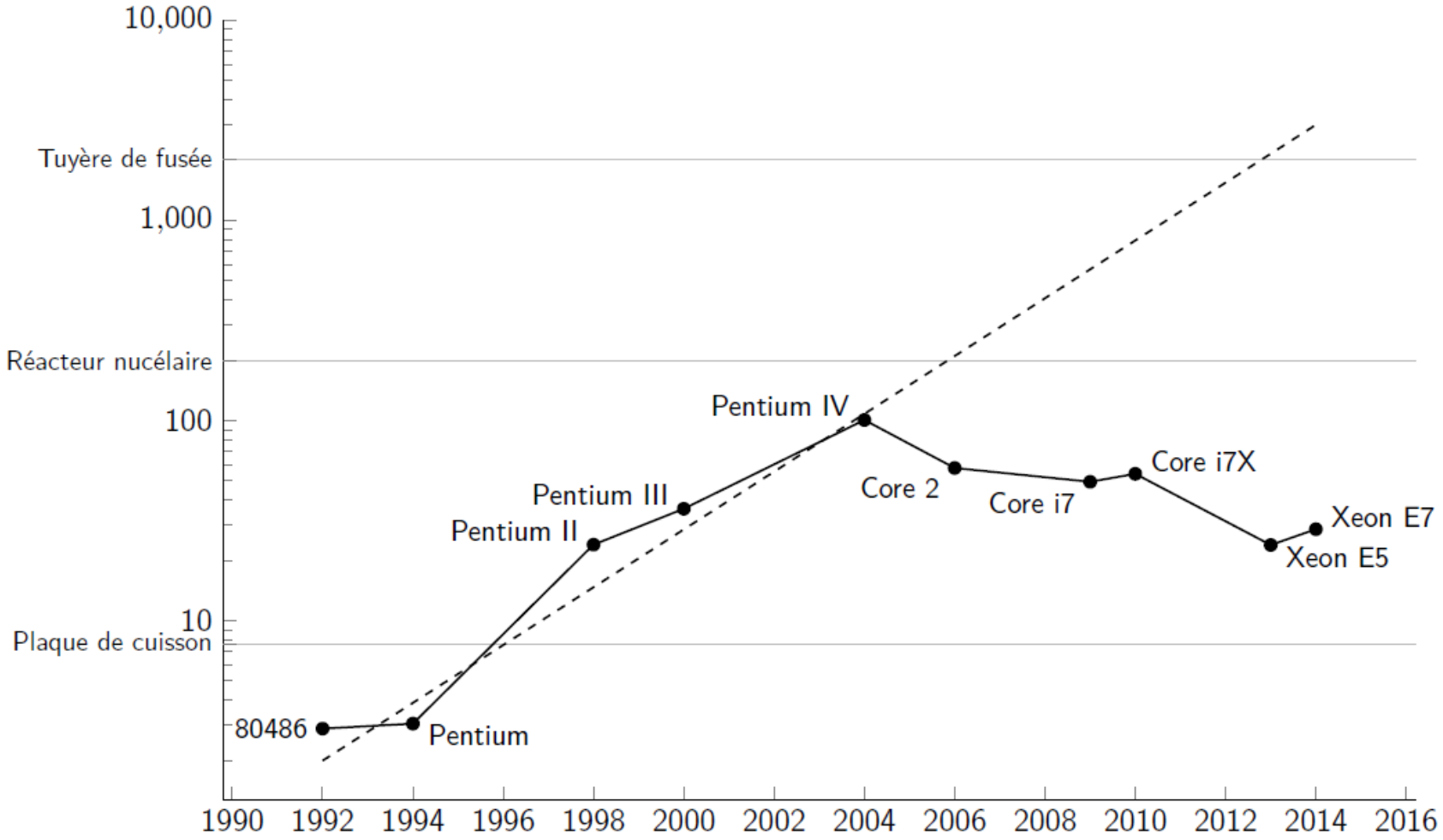
A la compensation de fréquence par parallélisme



Source: cours Intro Sys Emb ENSMA
Henri Bauer

Fréquence et dissipation thermique

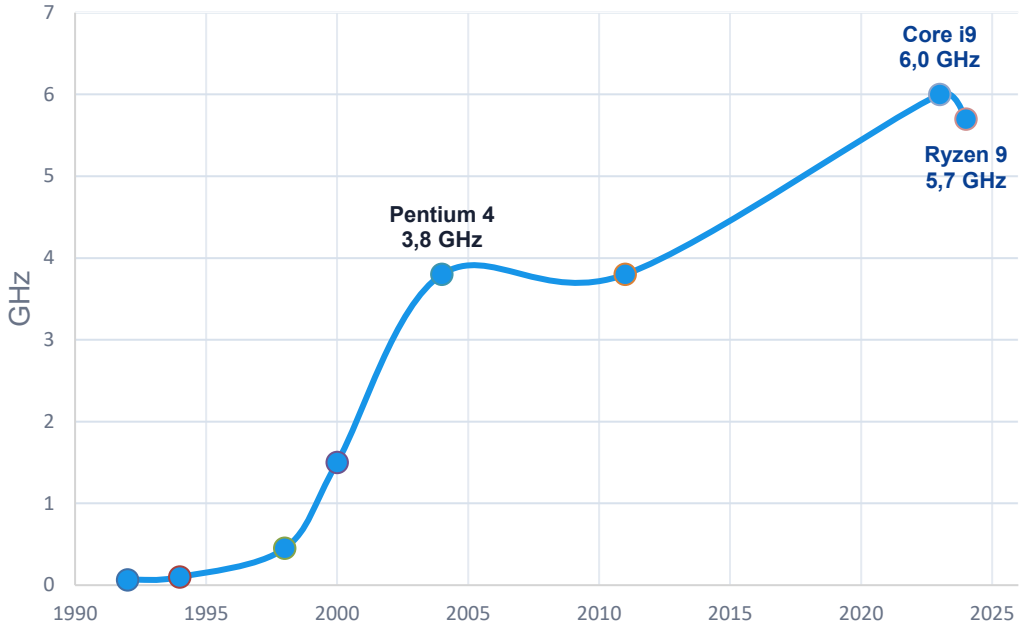
Évolution de la densité de puissance surfacique (en W/cm²)



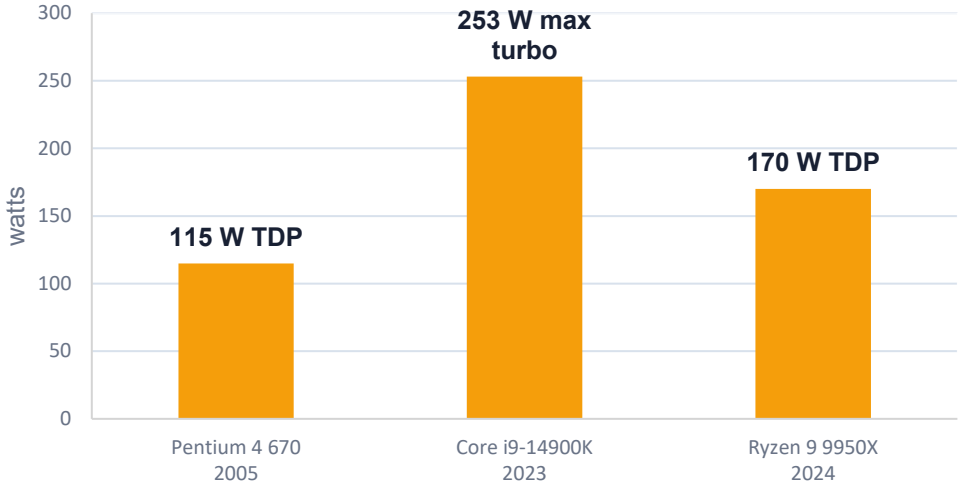
Source: cours Intro Sys Emb ENSMA
Henri Bauer

Le mur énergétique et thermique

Fréquence maximale représentative



Puissance thermique de référence du boîtier



Hyperthreading

- Lorsque les instructions et données sont en cache (et les données, qui sont produites, y seront jusqu'à ce qu'elles y soient remplacées), la mémoire est disponible et un autre « thread » pourrait utiliser la partie du processeur qui communique avec la mémoire pour commencer à charger ses instructions
- Le SMT (Simultaneous Multithreading) permet à plusieurs threads matériels de partager les ressources d'un même cœur pour mieux les utiliser. Il peut augmenter le débit global, mais introduit des interférences entre threads.

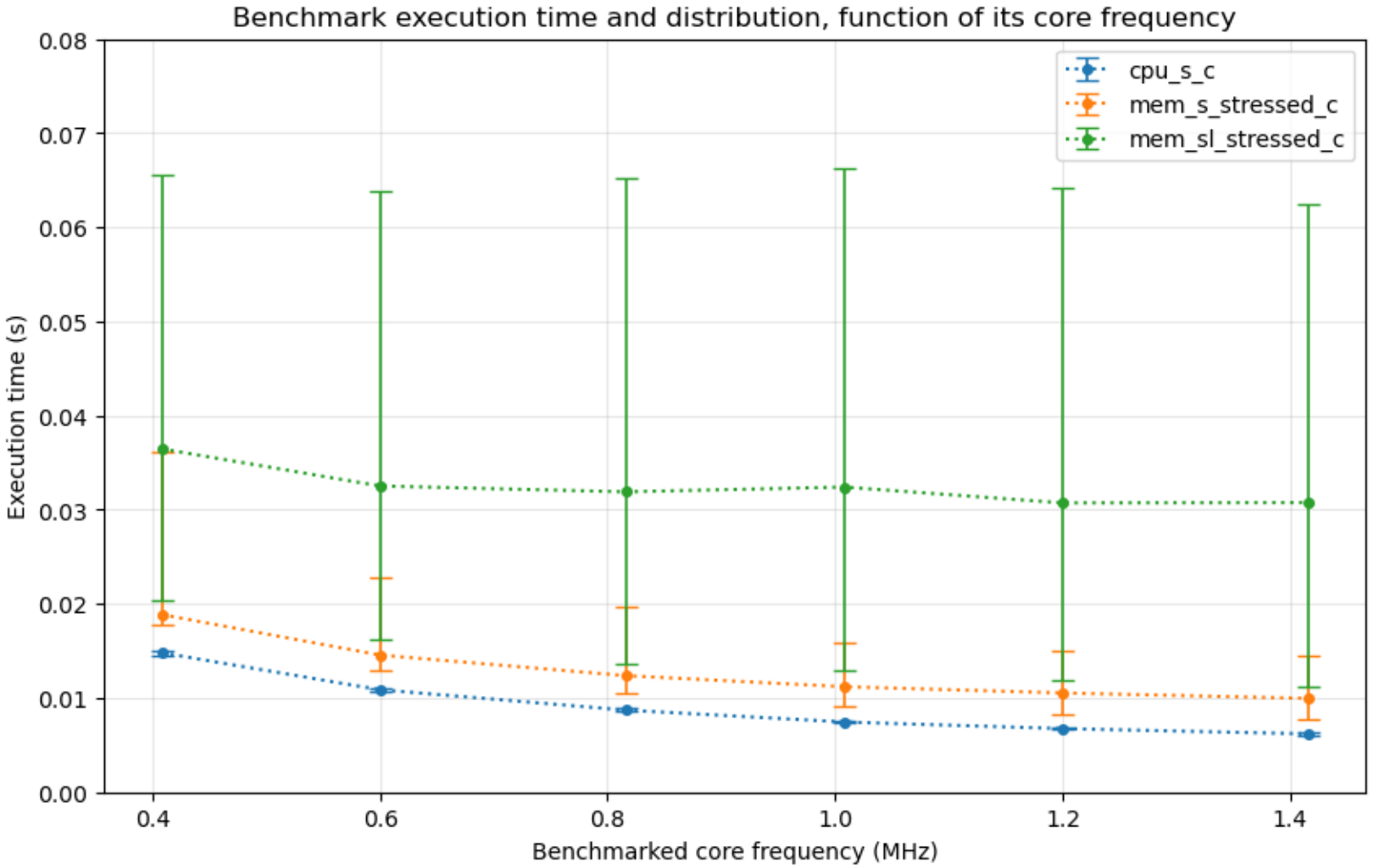
Ordonnancement de tâches

- Lorsqu'un programme est lancé, il est chargé dans la mémoire centrale (RAM), et un processus est créé
- Chaque processus contient un ou plusieurs fils d'exécution (threads en anglais, ou tâches)
- Un cœur exécute une tâche à la fois (ou 2 si il contient deux threads logiques)
- Une tâche n'est pas parallélisable sauf portions de code écrits dans des langages spécifiques (type CUDA pour exécution sur GPU de calculs type SIMD)
- Sur cœur de processeur, à chaque instant d'ordonnancement (tâche activée, tâche exécutée qui se bloque, fin de quantum de temps si ordonnanceur dirigé par le temps), l'ordonnanceur choisit la tâche à exécuter sur le cœur
- Que se passe-t-il sur plusieurs cœurs?

Ordonnancement multicoeur

- En général, chaque cœur possède une file d'attente de tâches à exécuter
- Périodiquement, ou lorsqu'un cœur est sur le point de se mettre en veille car inoccupé, ou lorsqu'un cœur surgit une charge élevée
  - Balance de la charge
  - Processus complexe consistant à affecter des tâches à d'autres cœurs
    - ✓ En 2016 un article montre que depuis des années, l'ordonnanceur Linux souffre de bugs liés à cela (voir https://blog.acolyer.org/2016/04/26/the-linux-scheduler-a-decade-of-wasted-cores/)
  - Lié à la gestion de fréquences et extinction des cœurs
- En programmation, on peut orienter ces choix en donnant une affinité à une tâche

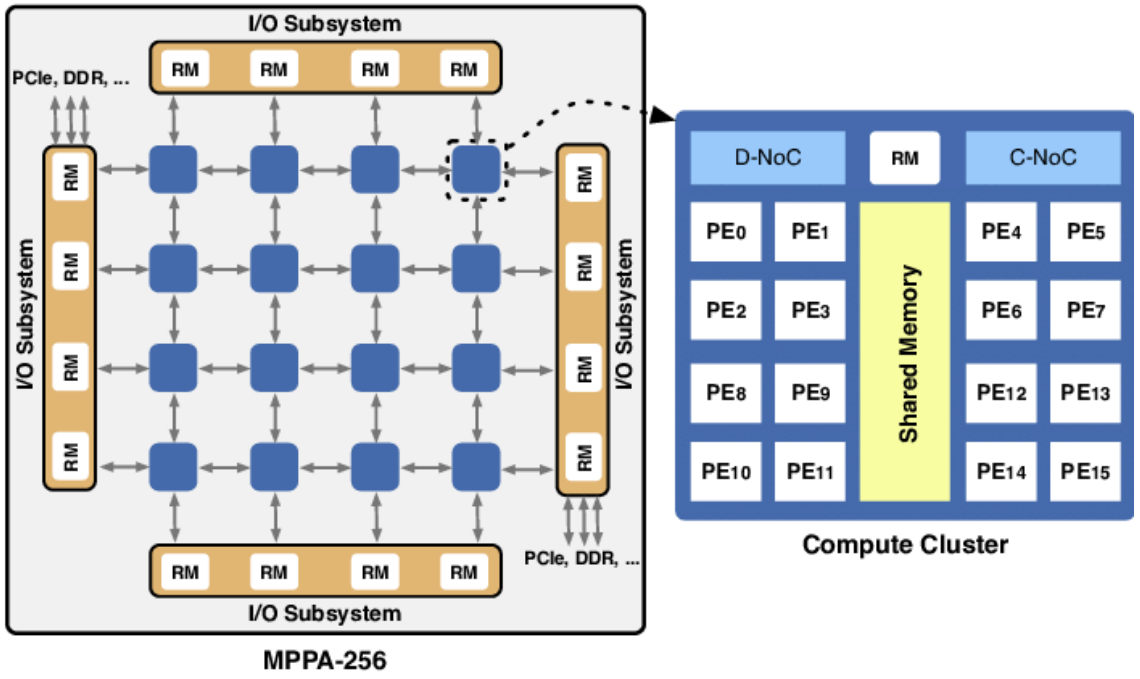
Des comportements moins prédictibles



Contention multicoeur

- Toutes ces architectures présentent de nombreux points de contention
  - Ressources internes de cœurs
  - Caches
  - FSB
  - Mémoire centrale
  - Etc.
  - Erreurs transitoires micro-architecture
- Risque d'utiliser 5% de la puissance ?
  - Criticité mixte? Probabiliste? …
- Eliminer la contention en pré-choisissant les instants d'accès ?

Manycore



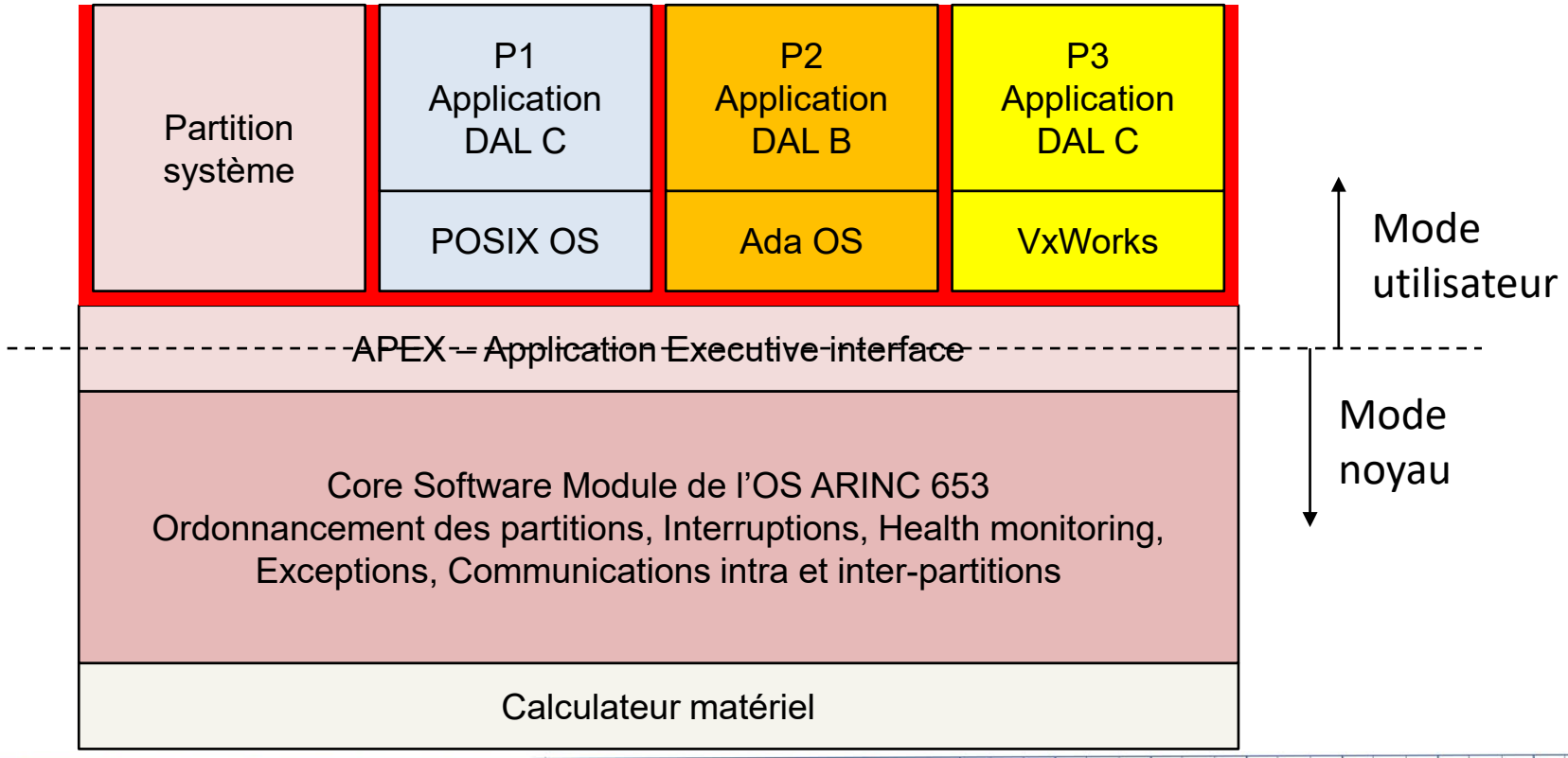
- Architecture complexe mais à très faible consommation
- Pourrait permettre de la dégradation gracieuse ?
- Moins sujette aux erreurs de micro architecture ?

Discussion évolution matérielle

- Des cœurs plus rapides mais qui consomment plus et dissipent sous forme de chaleur
- Depuis 2005 fréquence augmente peu mais nombre de cœurs augmente
- Nombreux problèmes de contention
  - Cache joue un rôle de plus en plus grand
  - Ordonnancement influe sur cache
  - Cache influe sur durée et donc ordonnancement
  - Analyse WCET et ordonnancement appelés à être plus liés

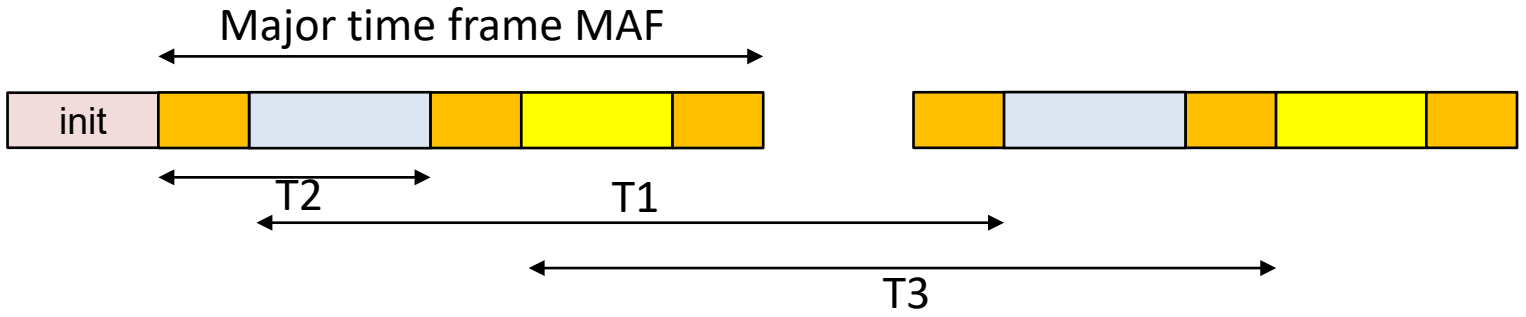
Le multicœur a-t-il sa place dans le critique ?

- CAST32A tente de définir des règles
- Utilisation dans l'avionique (AC 20-193 2024)

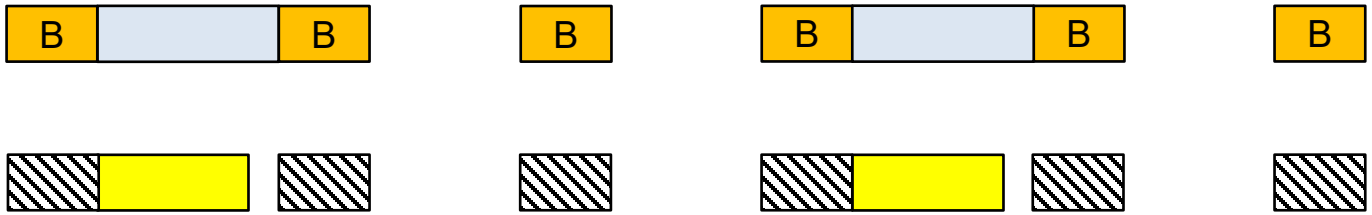


Premières utilisations du multicoeur

- Calculée statiquement



- Certains DAL requièrent l'arrêt des autres cœurs (en mode « trancheuse à jambon »)

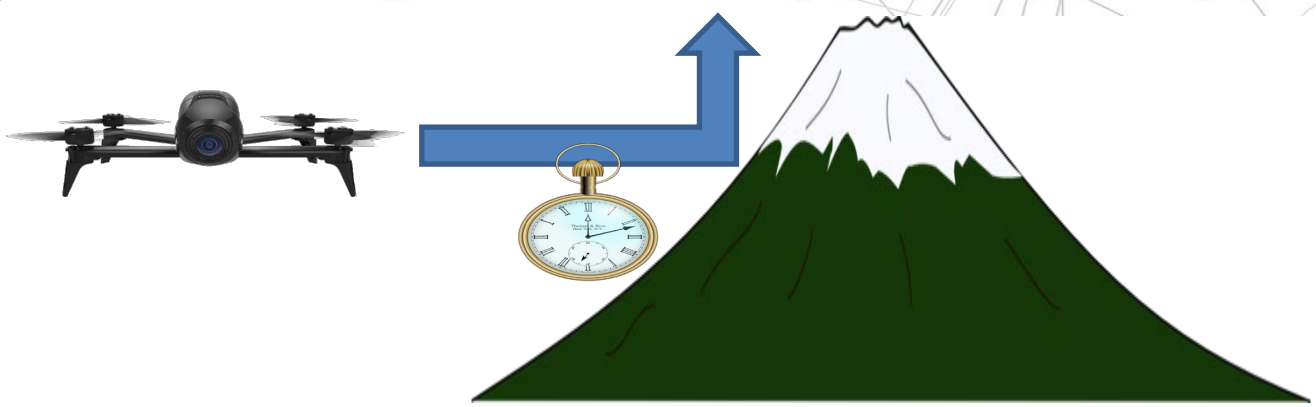


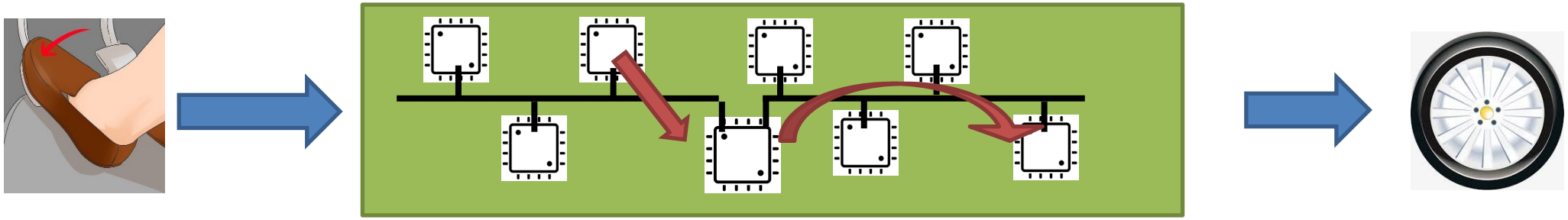
Mais d'où ça vient r, C, D, T, et ce modèle de Liu&Layland, ça représente quoi ?

3. ARCHITECTURE LOGICIELLE DES SYSTÈMES EMBARQUÉS TEMPS RÉEL

Contraintes de temps

- Définition 1990's
- Aujourd'hui contraintes de bout-en-bout
  - De ressentir à agir
  - Des capteurs aux actionneurs

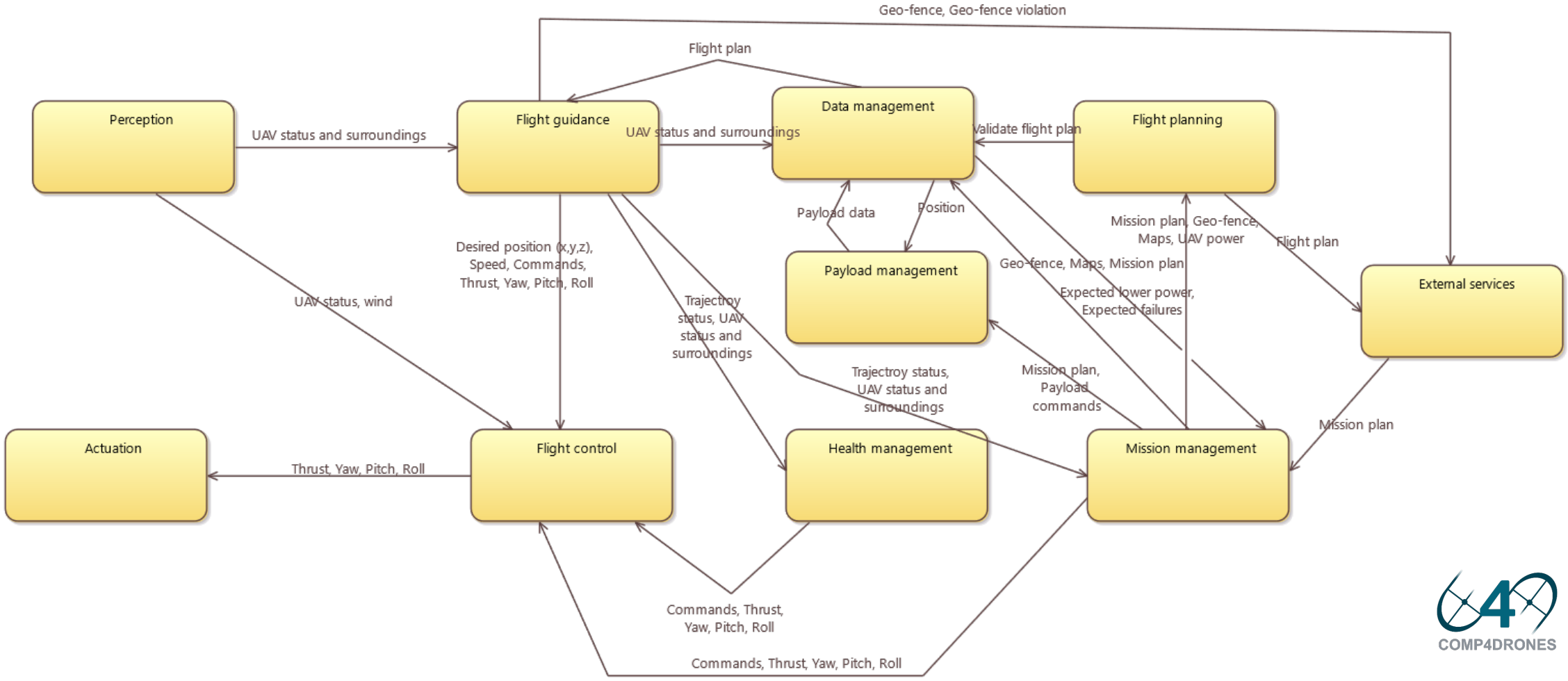




Conception de systèmes TR

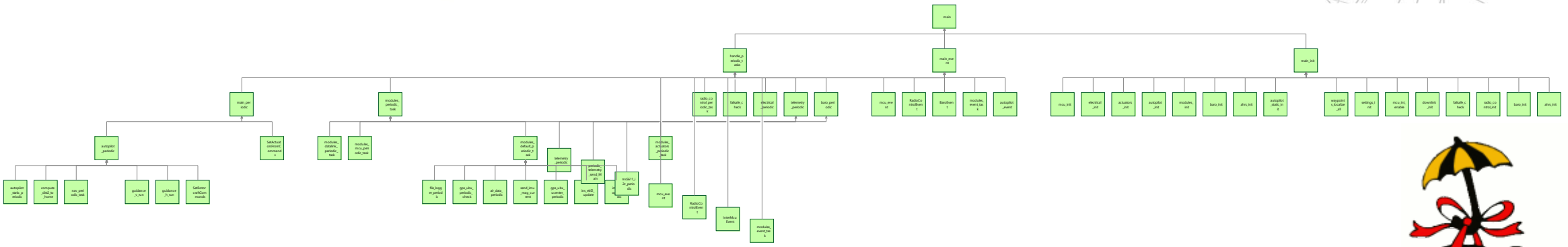
- Nombreux exemples de méthodes de conception
  - Arcadia, AUTOSAR, etc.
- Fonctions de haut niveau (~Use Case)
- Décomposition fonctionnelle (flots de données)
- Mapping sur architecture logicielle (threads, processus, OS, partition) et matérielle (calculateurs, réseaux)
- Nouvelles exigences apparaissent

Fonctions de haut niveau drone



Source: ECSEL H2020 Comp4Drones

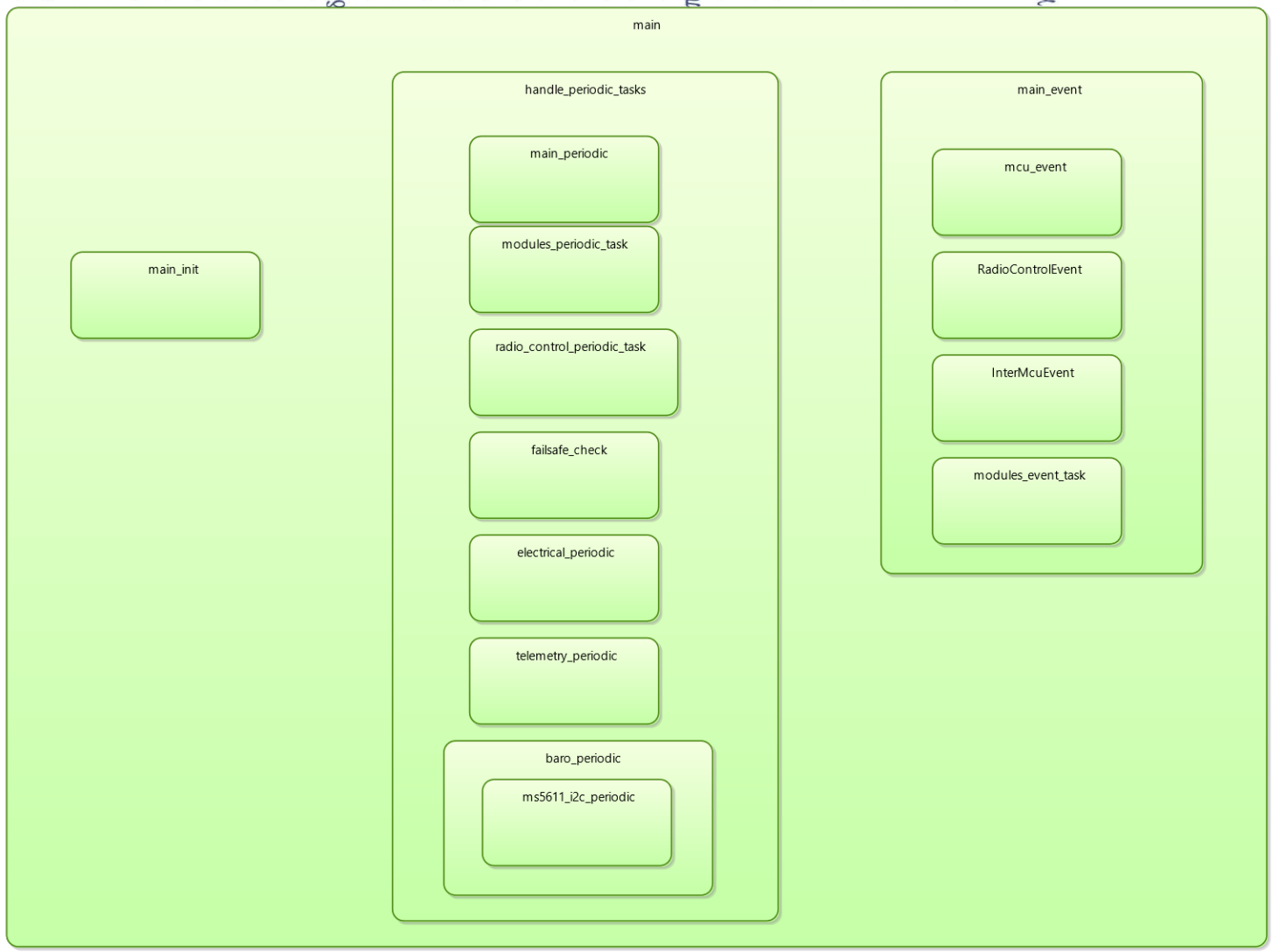
Autopilote Paparazzi



paparazziuav.org

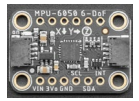
- Implémentation sur OS ou baremetal
- Deux premiers niveaux de décomposition fonctionnelle
- Ne représente pas les fonctions liées aux drivers

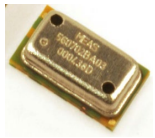
Représentation hiérarchique flot de données



Exemple drone bebop 2

- Calculateur dual core Cortex A9 + GPU + 8GO mémoire flash
  - Linux SMP PREEMPT
- Capteurs
  - Inertie
    - ✓ Magnétomètres 3-axes (AKM 8963) **I2C-1**
    - ✓ Gyromètres et Accéléromètres 3-axes (MPU6050) **I2C-2**
  - Vitesse sol
    - ✓ Caméra verticale capteur flux optique (∀16 ms compare images) **I2C-0**
  - Position
    - ✓ GNSS Ublox Neo M8N (GPS et Galileo et (GLONASS ou BeiDou)) **UART**
      - Envoi de trames réglable entre 1 et 30Hz
  - Altitude
    - ✓ Baromètre MS5607 **I2C-1**
  - Basse altitude
    - ✓ Sonar **SPI** pour déclencher lecture, **ADC** pour lire





Actionneurs, horloges, charge utile

- Module WiFi
- GPIO
  - Alimentations diverses
  - Bouton On/Off
- PWM
  - Résistances chauffantes IMU
  - Horloges pour capteurs gyro&accéléro
  - Horloges pour les caméras
- Commande moteurs hélices
  - BrushLess Driver Controller (BLDC) **I2C-1**

Différents types d'E/S capteurs/actionneurs

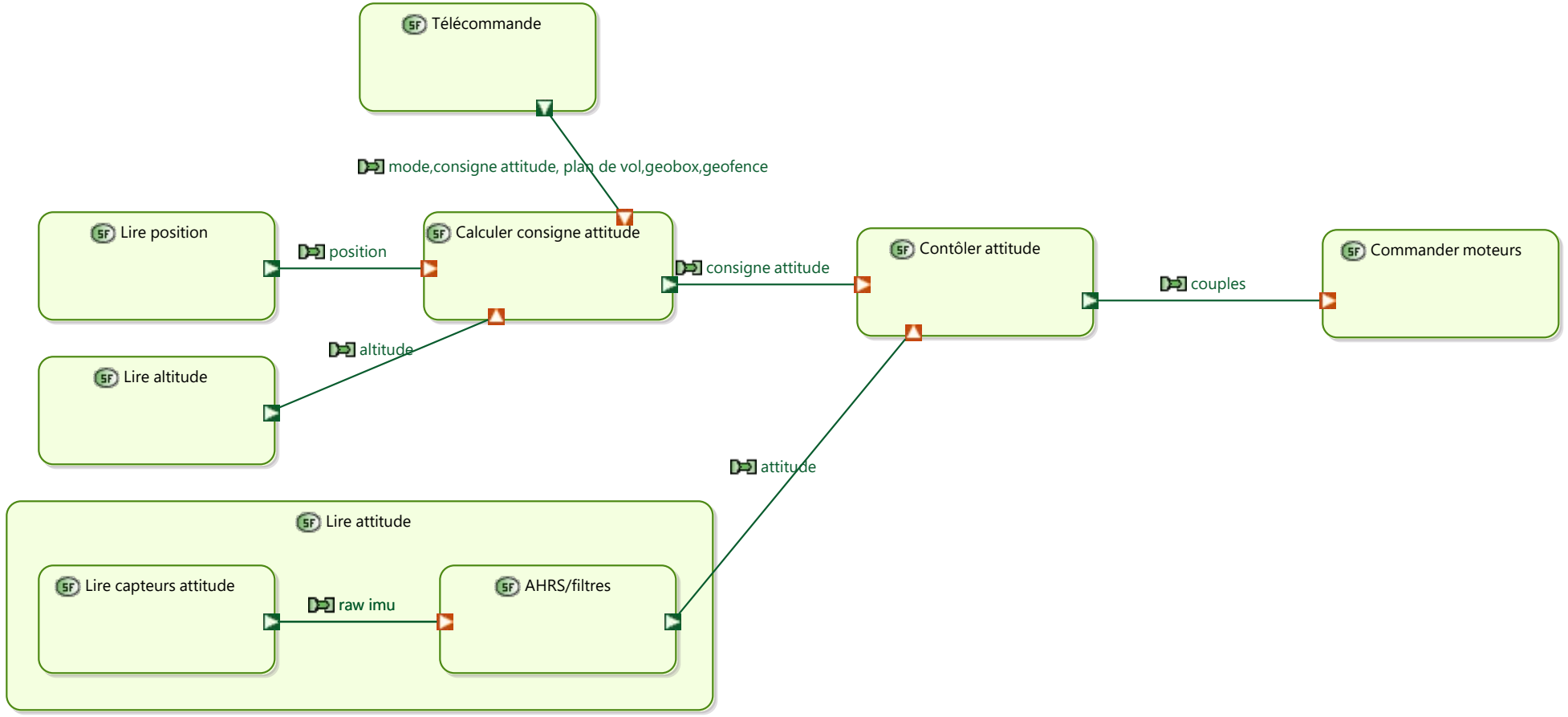
- Créent différentes contraintes (notamment temporelles)
- Autorisent ou non différents modes de réaction logicielle
  - Génération ou non d'interruption ⇒ scrutation ou événement
  - Mode push, pull, ou requête-réponse
    - ✓ Push : une ISR, voire une tâche peut être déclenchée qui peut déclencher une chaîne de tâches
      - Mais si rien n'arrive ?
    - ✓ Pull : obligé de scruter, plus on scrute vite, plus on perd du CPU, plus on scrute lentement, plus on perd en réactivité
    - ✓ Requête-réponse : suspension, perte de temps CPU si pas multitâche, problème NP-difficile au sens fort pour l'ordonnancement

Types d'E/S vs modèles de programmation

	Evénement (ISR) et si oui élément requis	Scrutation	Requête réponse
GPIO (E/S numériques)	Certaines E/S numériques seulement	**Oui**, 1 E/S = 1 bit	Non
ADC (entrées analogiques)	Non sauf exception coûteuse	**Oui**, sachant qu'un ADC possède une fréquence (centaines de kSamples/s)	Non
UART (généralement en comm asynchrone)	Si contrôleur série, **oui** à chaque octet reçu	Pilote typique bufferise les octets sur ISR, **scrutation sur le buffer**	Possible mais lent en bi-directionnel, rare
I2C et SPI (bus synchrones de type maître-esclave)	**Oui** en une phase requête, une phase lecture	**Oui**, si octets reçus bufferisés par ISR	**Oui**, réponse envoyée de façon synchrone par le périph interrogé dans la trame envoyée par un maître
PWM	**Idem GPIO pour lecture, nécessite horloge programmable pour écriture**	Non sauf si prêt à utiliser un cœur entier	Non
Bus de comm (CAN, Ethernet, WiFi, etc.)	**Oui s'il existe le contrôleur adéquat**	**Pilote typique bufferise les trames sur ISR, scrutation sur buffer**	Non

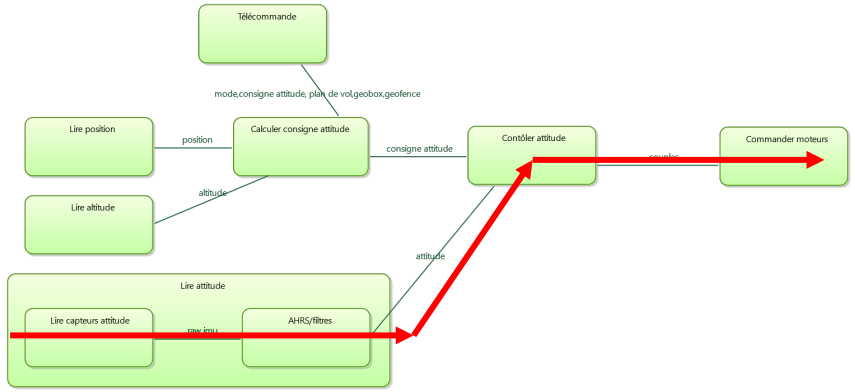
Archi logicielle et sûreté de fonctionnement

❑Extrait d'un diagramme fonctionnel flots de données



Régulation de l'attitude

- Le contrôleur d'attitude doit s'exécuter au moins à 400 Hz
  - Période ≤ 2,5 ms



Chaîne fonctionnelle
Régulation attitude

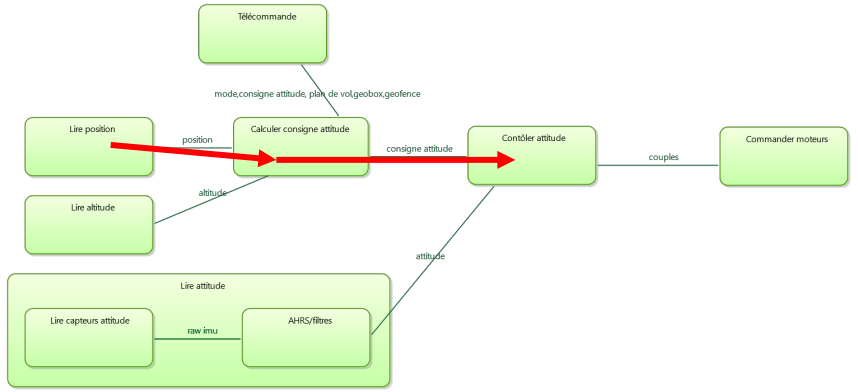
- Interroger et lire via I2C 3-accél, 3-gyros, température
  - 1 octet écrit, 14 octets lus
  - ~10 bits par octets à 400kHz (fréquence bus I2C) :
    - ✓ 150/400000 = 0,375 ms (15% de 2,5 ms)
- Interroger et lire via I2C 3-magneto
  - ~0,225 ms (9% de 2,5 ms)
- Filtrage et AHRS parfois gourmand
    - ✓ Exemple : calculs de matrice quaternions
- Mode requête réponse (tâche qui se suspend) risque de fortement nous contraindre

Lecture I2C classique Paparazzi

- ISR bufferise octets I2C reçus
- Itération i
  - Envoi requête i
  - Lecture buffer (données correspondent à la requête i-1)
  - fonction de traitement des données i-1
- Toujours un coup de retard
- Nécessite qu'un temps suffisant s'écoule entre envoi requête i-1 et lecture buffer i
  - Ici ~400µs
  - Sinon, utilisation de données vieilles d'au moins 2 itérations
- Nécessite de ne pas remplir le buffer I2C avant lecture

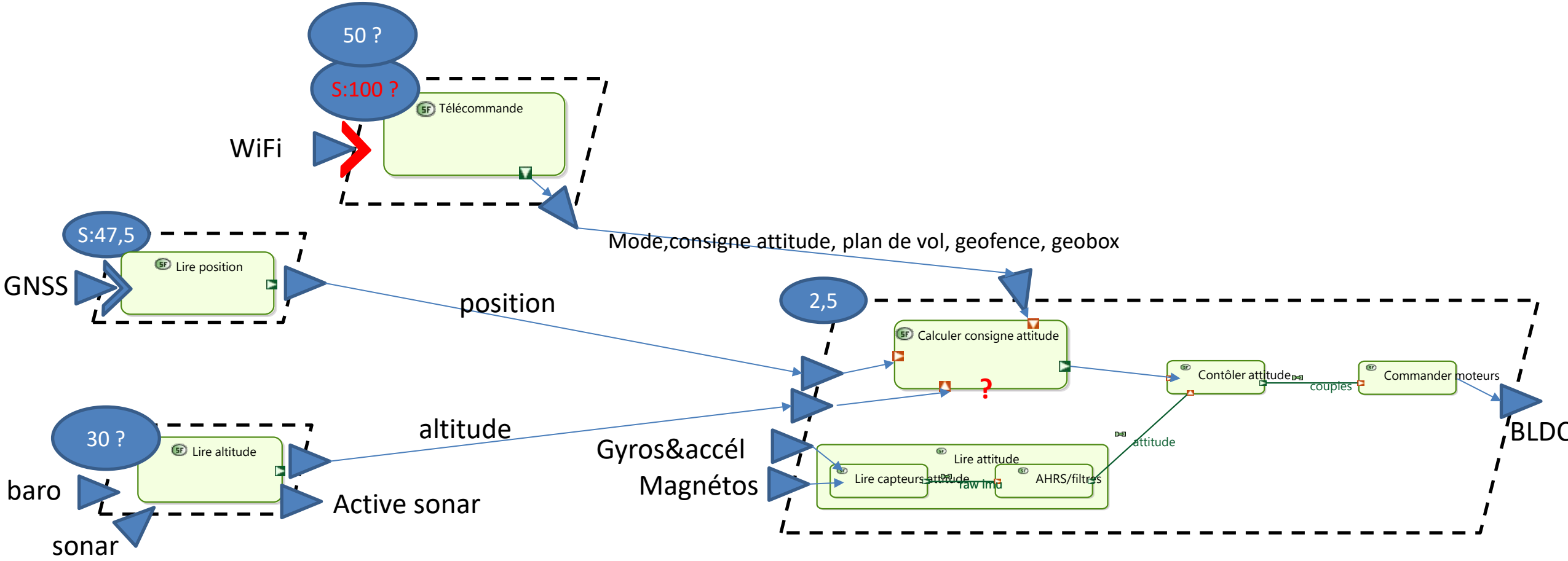
Chaîne fonctionnelle geofencing

- Supposons que si trop proche barrière
  - Changer trajectoire&consigne attitude
- Supposons GNSS réglé à 20 Hz
- Période arrivée trames GNSS : 50 ms+/-5%
  - Délai min inter arrivée 47,5 ms



Chaîne fonctionnelle
Evitement obstacle sol

Choix architecture logicielle



- Exemple exigence NF : Le délai maximum entre l'arrivée dans une position proche d'une geofence/geobox et sa prise en compte pour la commande moteur ≤ 100 ms

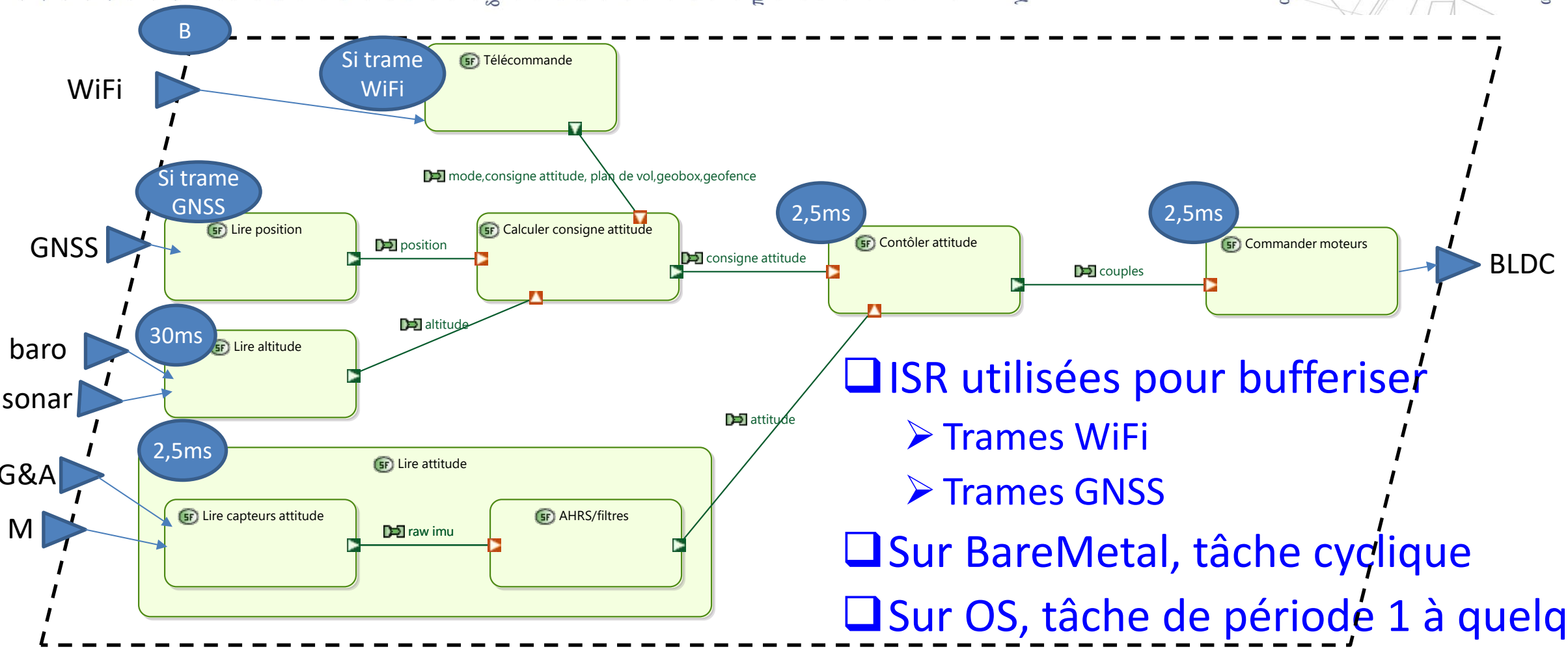
Contraintes de temps

- Contrainte de bout-en-bout
  - délai GNSS + temps de réponse tâche lecture position + (période + temps de réponse tâche commande de vol) + délai action BLDC sur moteurs
- Contraintes de délai critique des tâches
  - Souvent ré-entrance des tâches indésirable et délai critique ≤ période ou échéance sur requête
- *Event based* plus réactif car synchronisé sur événement
- Mais tolérance aux fautes plus complexe à maîtriser
  - Si liaison station sol rompue, tâche télécommande ?

Plus ou moins de tâches ?

- Plus de tâches
  - \+ Plus tirer parti du parallélisme des calculateurs
  - \+ « Facilement » réparti sur plusieurs plateformes
  - \+ Redondance plus simple
  - \+ Plus modulaire
  - – Délais de bout-en-bout calculables en temps acceptable plus longs
  - – Plus d'overhead, de préemptions (donc impact cache)
  - – Plus de complexité à analyser, valider, rendre consistant
  - – Plus de mémoire car variables locales, fifo de communication, outils de synchronisation et
    - ✓ Σmax piles fonctions > max(piles fonctions)

Architecture monolithique drone



- ISR utilisées pour bufferiser
  - Trames WiFi
  - Trames GNSS
- Sur BareMetal, tâche cyclique
- Sur OS, tâche de période 1 à quelques ms

De plus en plus de problèmes sont à traiter

- Systèmes de plus en plus grands et complexes, qui sont globalement asynchrones, mais localement synchrones
- Plateformes de plus en plus performantes mais complexes
  - Nombreux points de contention
  - Nombreux caches, utilisation dépendante de l'ordonnancement
  - Erreurs transitoires
  - Différence durée moyenne vs. WCET de plus en plus difficile à contenir
  - Nouvelles plateformes et nouveaux usages (ex: GPU et IA)
- Modèles d'application et de plateforme trop simplistes
  - Choix de modèles plus fins et représentatifs
  - Nouveaux paradigmes à explorer pour compenser l'écart WCET vs. grande majorité des cas (MCS? Probabiliste? Ordo analyse WCET conjoints? Modèles fins d'accès mémoire? etc.)
  - Approches diminuant l'impact des points de contention
  - Remonter ce qu'on peut de l'analyse de perfs dans le cycle de vie logiciel (MBSE ?)
- Nouveaux protocoles réseau pour le temps réel
- Dimension sécurité à développer (Cyber Resilience Act – EU 2024)

Merci, et bonne école d'été!!!