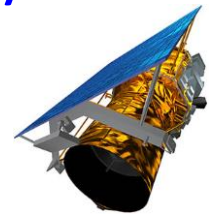
Temps réel

Emmanuel Grolleau

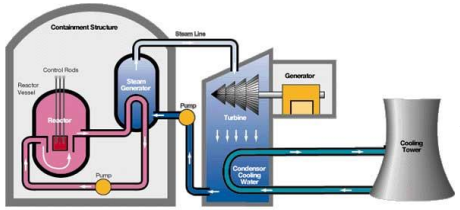
Définitions

- 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, UN BESOIN INDUSTRIEL

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
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

10^9h>114000 ans

- 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

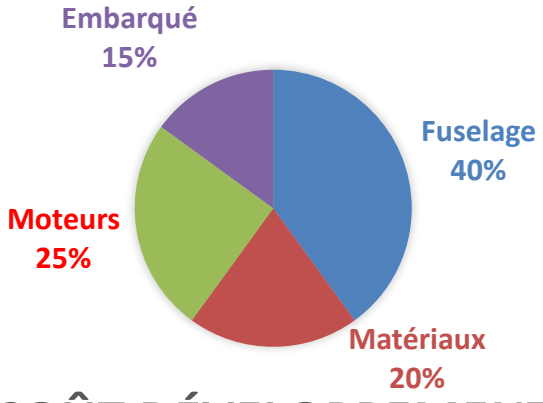
Mardi

Où se trouve le temps réel d'ETR ?

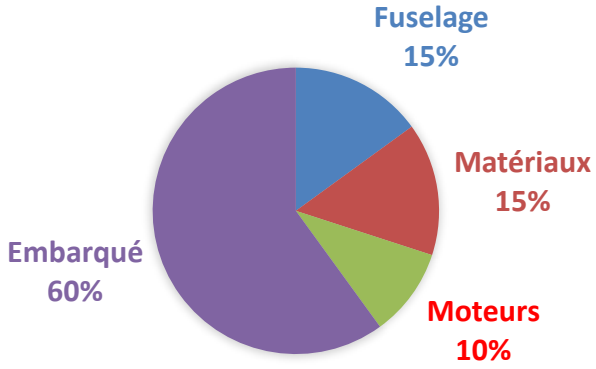
- Respect des échéances : exigences non fonctionnelles
- Exigences liées aux performances
  - Contraintes de bout-en-bout
  - Pouvant se décliner en échéances sur les tâches et les messages
- Dans la DO-178C, en particulier dans la phase *Verification of Outputs of Software Design Process*
  - Traçabilité 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)

Logiciel critique dans l'aéronautique

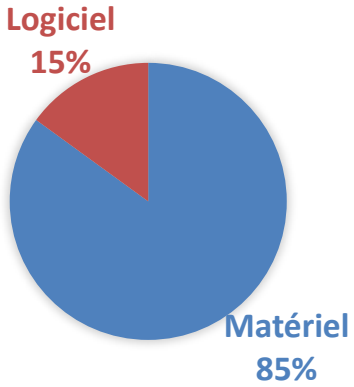
DURÉE DÉVELOPPEMENT 1960



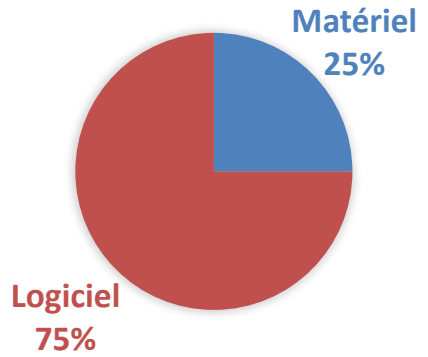
DURÉE DÉVELOPPEMENT 2010



COÛT DÉVELOPPEMENT 1980

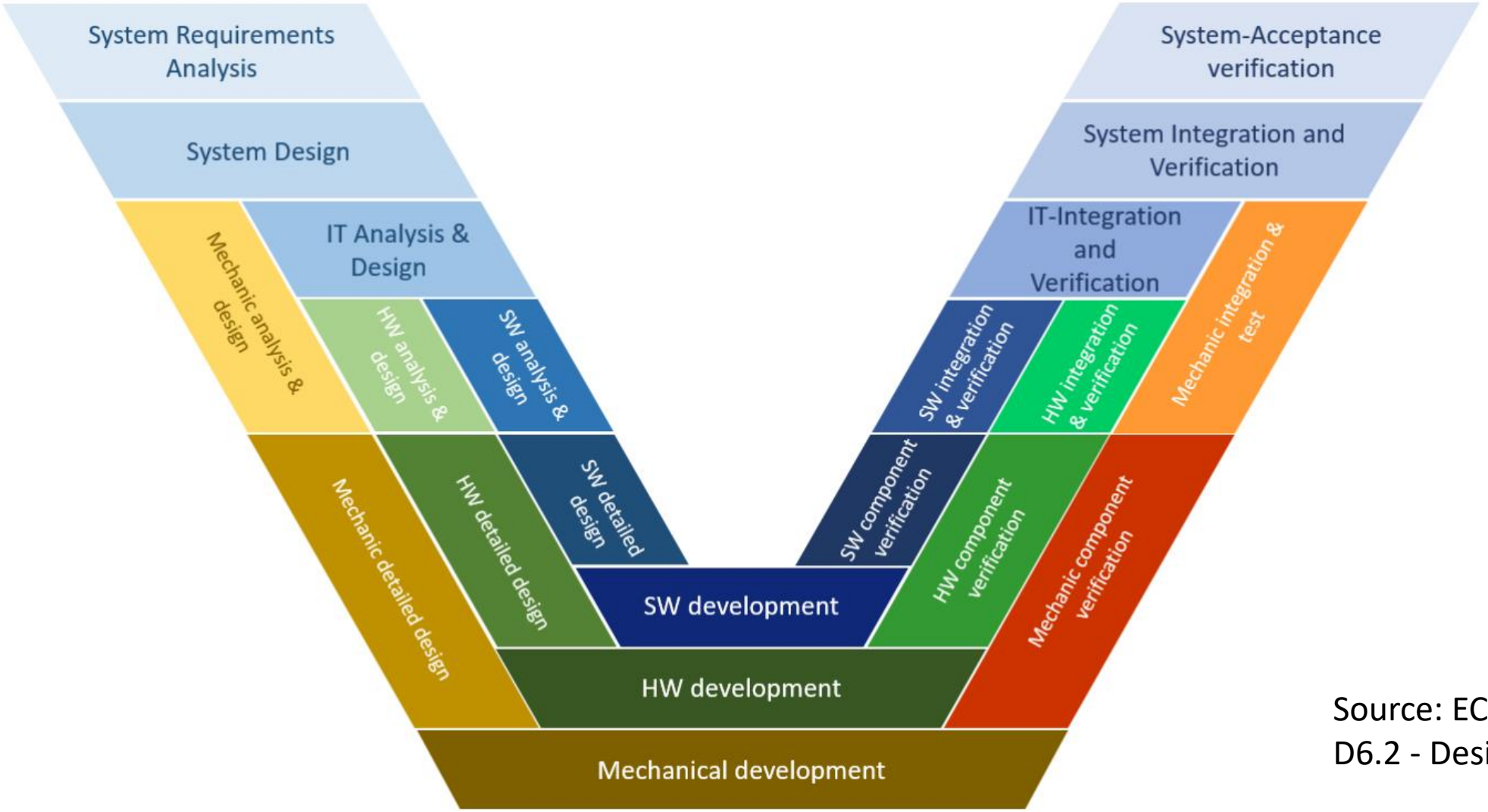


COÛT DÉVELOPPEMENT 2010



Source: NASA Glenn Research Center

Cycle en V



COMP4DRONES

Source: ECSEL H2020 Comp4Drones D6.2 - Design Tools

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
  - Exemple
- 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?
  - ✓…

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

2. DES PROCESSEURS OPTIMISÉS POUR LES PERFS MOYENNES

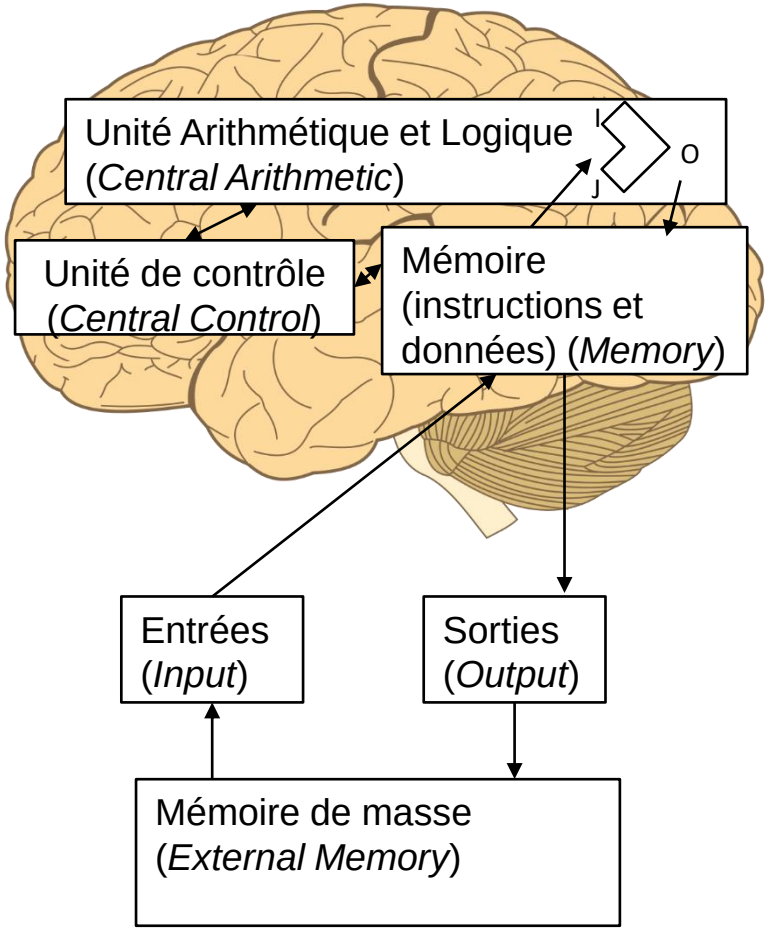
2.1 LE GOULOT D'ÉTRANGLEMENT DE LA MÉMOIRE

Architecture « théorique » von Neumann

- John von Neumann (1903-1957) – mathématicien Hongro-Américain
  - A théorisé les automates cellulaires

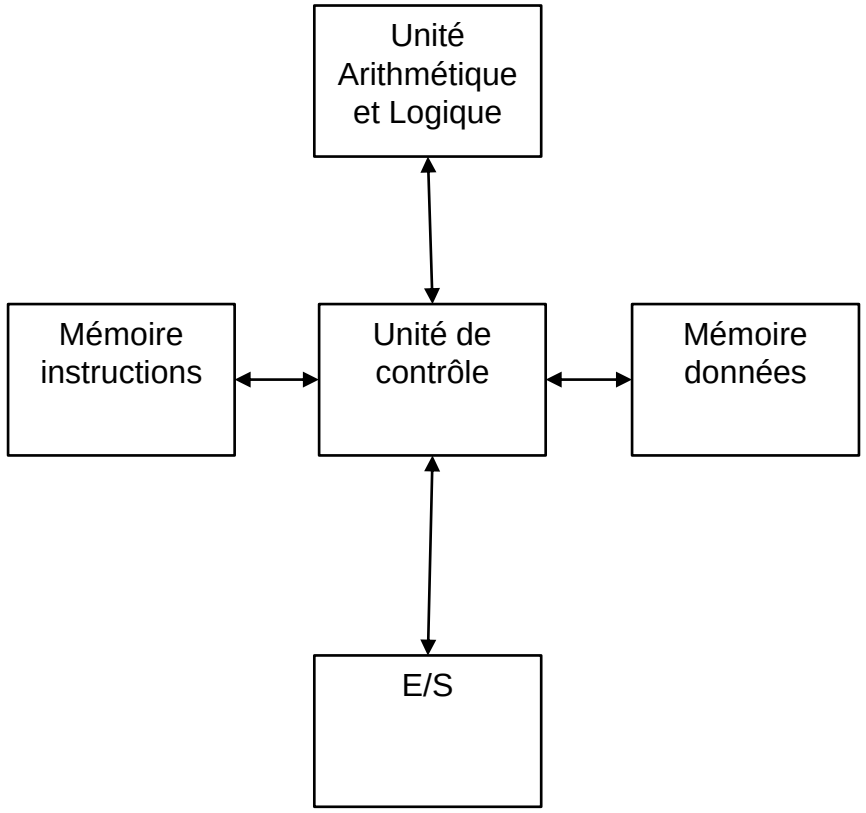
    https://youtu.be/CfRSVPhzN5M
  - A participé au projet Manhattan
    - ✓ A développé les premiers algos Monte Carlo sur l'ENIAC
  - A développé des logiciels de modélisation du climat
    - ✓ A averti en 1955 des risques de l'activité humaine sur le réchauffement climatique
  - Invente le 1er algorithme de tri (tri fusion) de tableau de complexité pire cas O(nlog(n)) en 1945 (le prochain sera inventé 20 ans plus tard)
  - Participe à proposer une architecture d'ordinateur pour l'EDVAC qu'il théorise en 1945

Architecture von Neumann

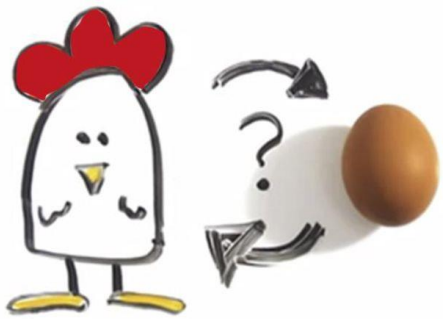


- Proposée en 1945 pour l'ENIAC
- Utilisation du binaire
- Agrégation de E-éléments (portes logiques)
- Utilisation d'une horloge commune
- Circuits séquentiels créant des registres

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



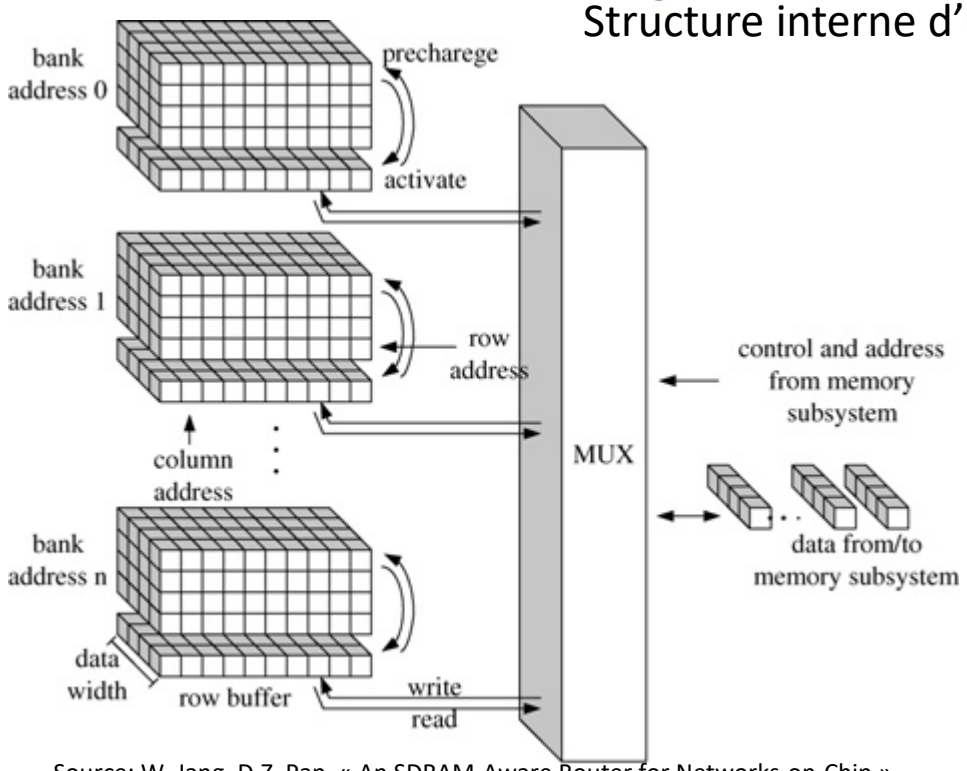
- Les ordinateurs modernes, von Neumann ou Harvard modifié ?

Calculateur quantique

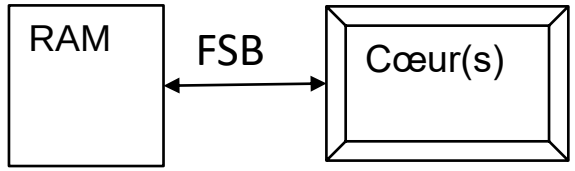
- Pas pour demain, et spécialisé dans la recherche/décomposition d'algorithmes traitant de façon exacte des problèmes NP
- Révolution de l'ordinateur ou circuit spécialisé type FPU ou GPU pour la résolution de problèmes NP spécifiques transférés par l'UC ?
- L'article Google et al. Co-signé par 77 auteurs paru dans Nature en octobre 2019 relate que la suprématie quantique a été atteinte
  - Pour ce faire, ils ont fait de l'échantillonnage de circuit quantique aléatoire sur 53 qubits, vs. une estimation du temps de calcul sur un supercalculateur actuel
  - 600 seconds vs. 50 milliard d'heures de calcul core sur Google cloud
- Devrait-on alors parler de GPU, de FPU, d'ASIC comme d'une autre famille de calculateur ?

Mémoire centrale

Structure interne d'une SDRAM



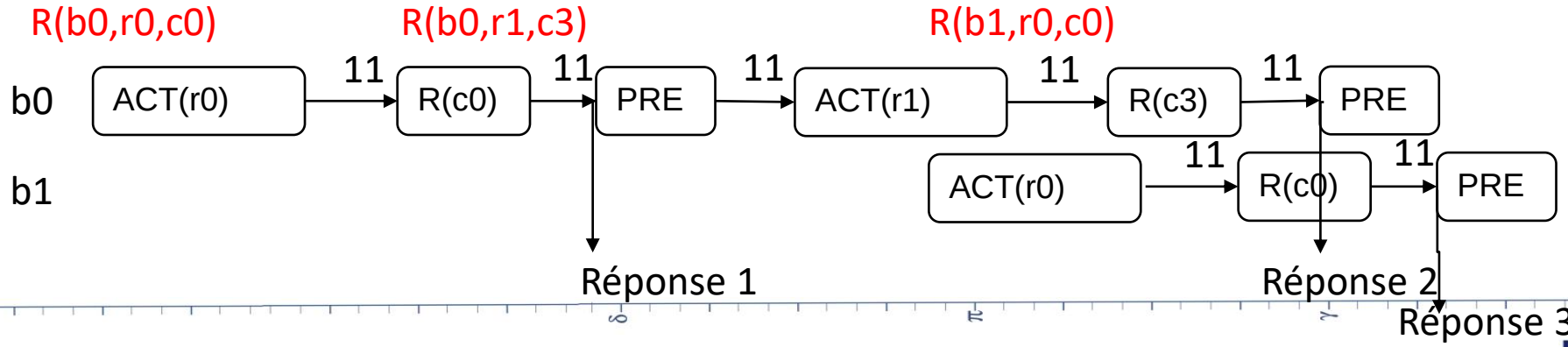
Source: W. Jang, D.Z. Pan, « An SDRAM-Aware Router for Networks-on-Chip »,
IEEE Tr. on Computer-Aided Design of Integrated Circuits, 29(10) 2010



- Avant de R/W une colonne, on doit Activate(bank,row)
  - R/W(bank, row, column)
- Délais nécessaires pour chaque opération
- Possibilité de paralléliser (en pipeline) des opérations sur des banques différentes

Mémoire centrale/suite

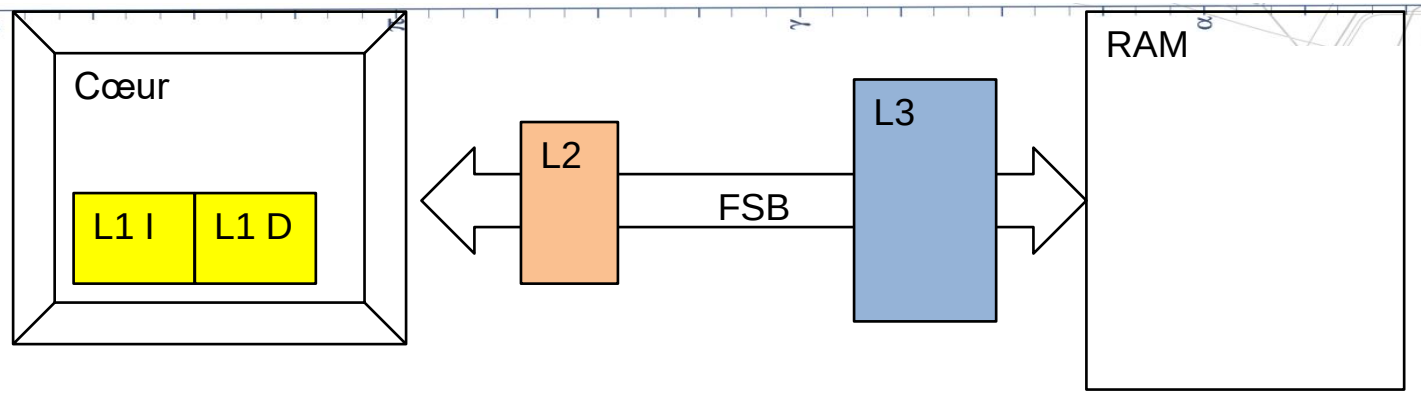
	DDR I SDRAM			DDR II SDRAM				DDR III SDRAM			
Freq (MHz)	**133**	**167**	**200**	**200**	**267**	**333**	**400**	**400**	**533**	**667**	**800**
Read	2	2,5	3	3	4	4	6	6	8	10	11
Write	1	1	1	2	3	3	5	5	6	7	8
Act	2	3	3	3	4	4	6	6	8	10	11
Column	1	1	1	2	2	2	2	4	4	4	4
Precharge	2	3	3	3	4	4	6	6	8	9	11
W-to-R	1	1	2	2	2	3	3	4	4	5	6
R-to-W	0	0	0	0	0	0	0	7	8	9	9
W recov	2	3	3	3	4	5	6	6	8	10	12



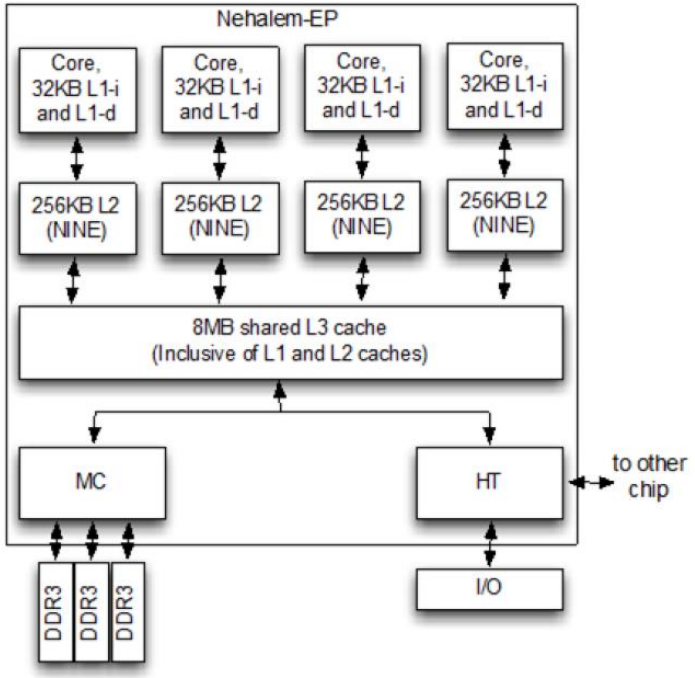
Mémoire centrale /suite

- 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
- 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 la mémoire
  - Exemple : 33 cycles d'attente entre deuxième lecture et réponse de la DDR III SDRAM à 800 MHz. Si processeur à 4 GHz, cela se traduit par 4*33/0,8=165 cycles du processeur! On ne compte même pas la contention d'accès au FSB
  - Lorsque plusieurs cœurs sont en concurrence pour accéder à la mémoire, cela devient encore plus complexe
  - Le cache mémoire joue un rôle très important de diminution des délais d'accès à la mémoire en faisant jouer le **principe de localité**.

Mémoire cache



- Plus le cache est rapide, plus il est cher, et plus il consomme d'énergie (rafraîchissements fréquents)

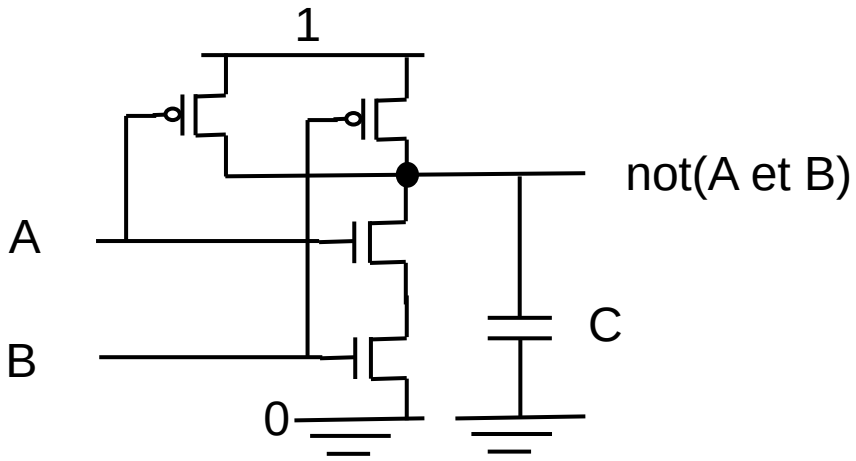


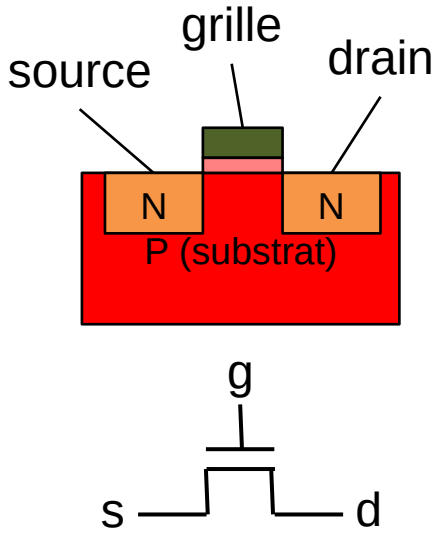
Niveaux de cache dans l'architecture multicoeur
Intel Nehalem (premiers Core i7)

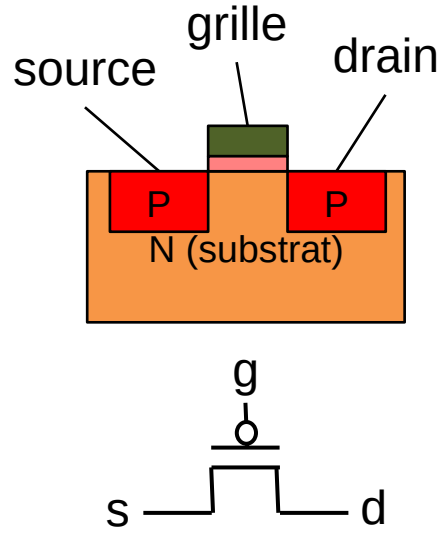
2.1 LA MONTÉE DU PARALLÉLISME

Transistors

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

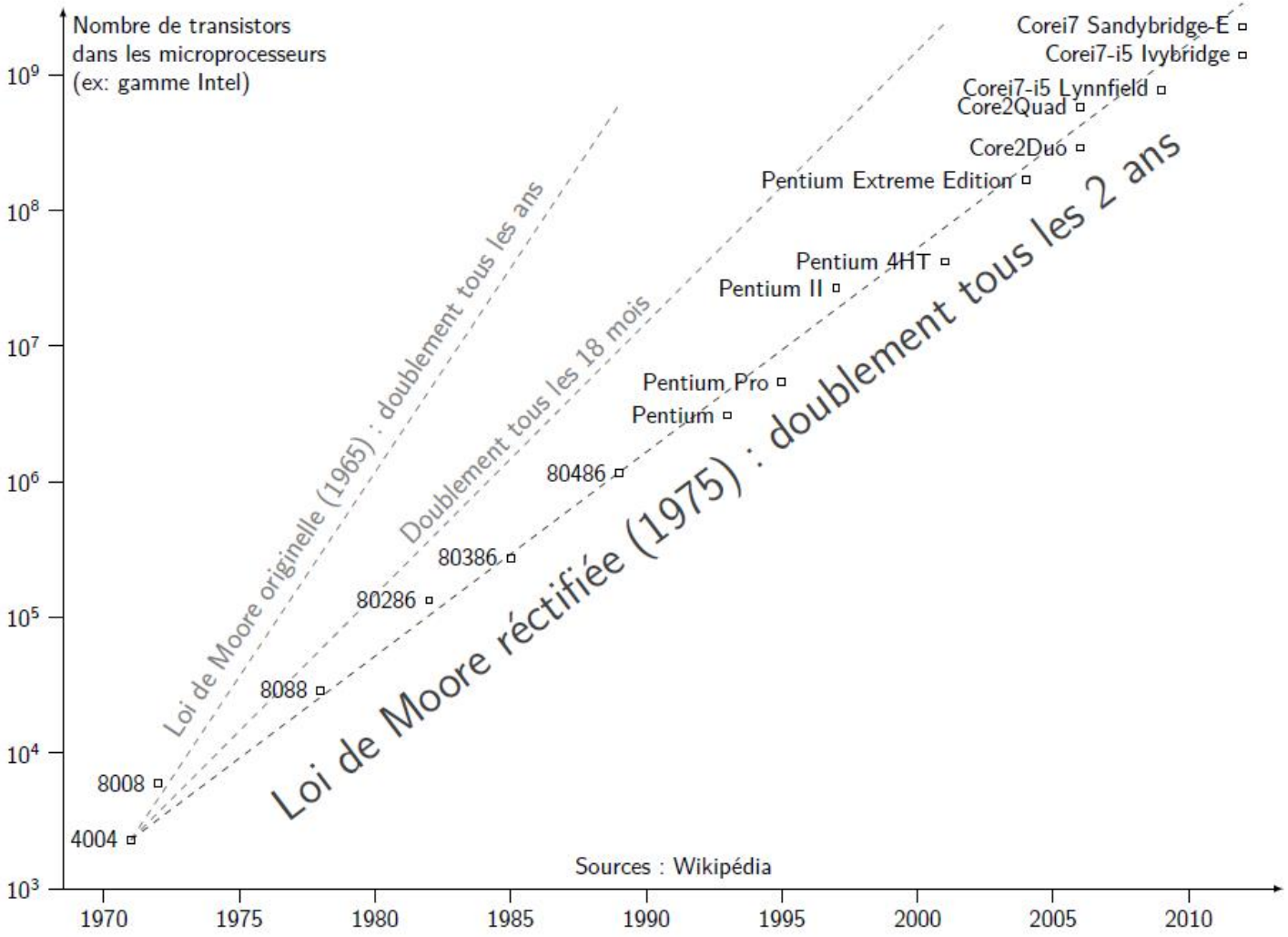






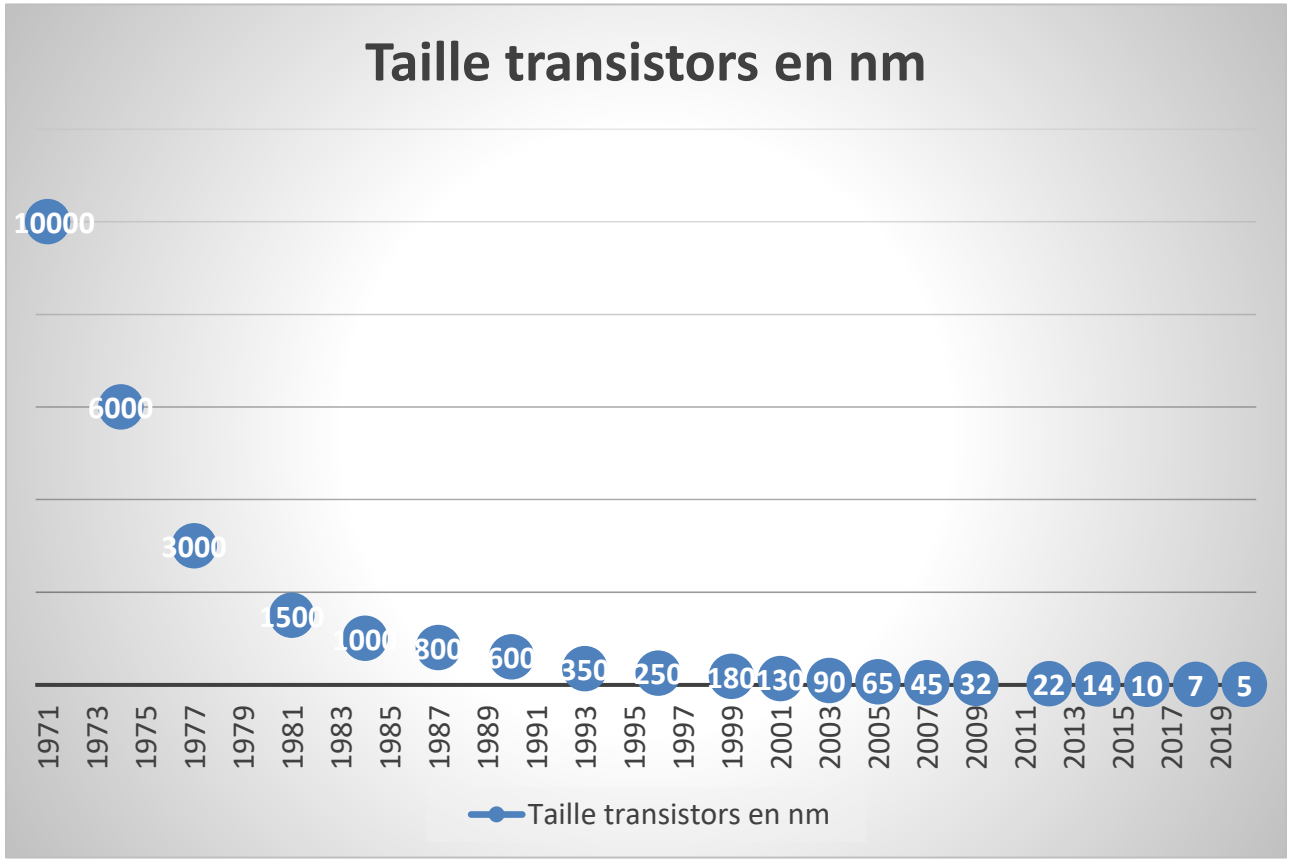
- Puissance consommée :
  - $P=fCU^2+P_{statique}$

Des promesses de la loi de Moore

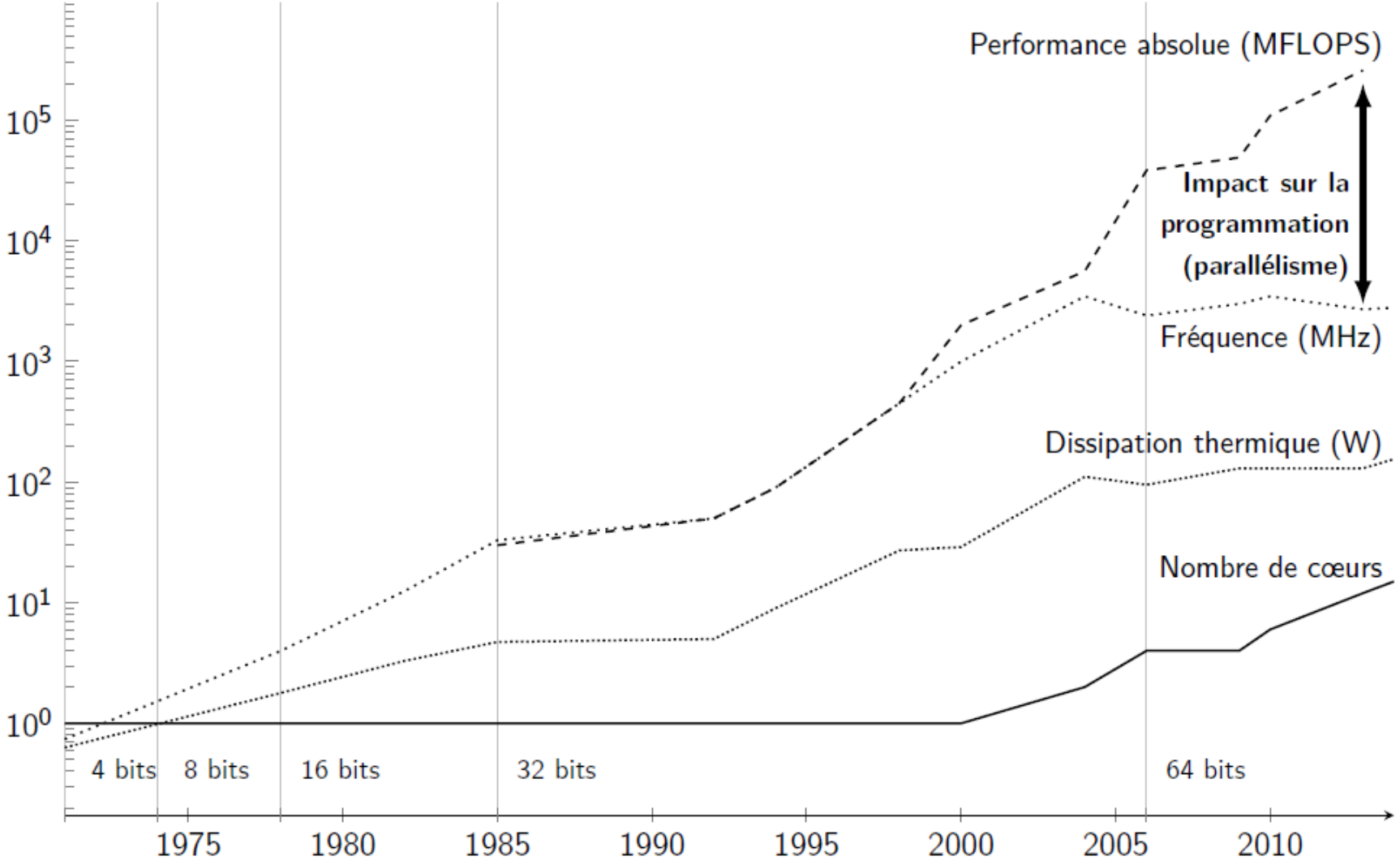


Source: cours Intro Sys Emb ENSMA
Henri Bauer

A la finesse « théorique » de gravure



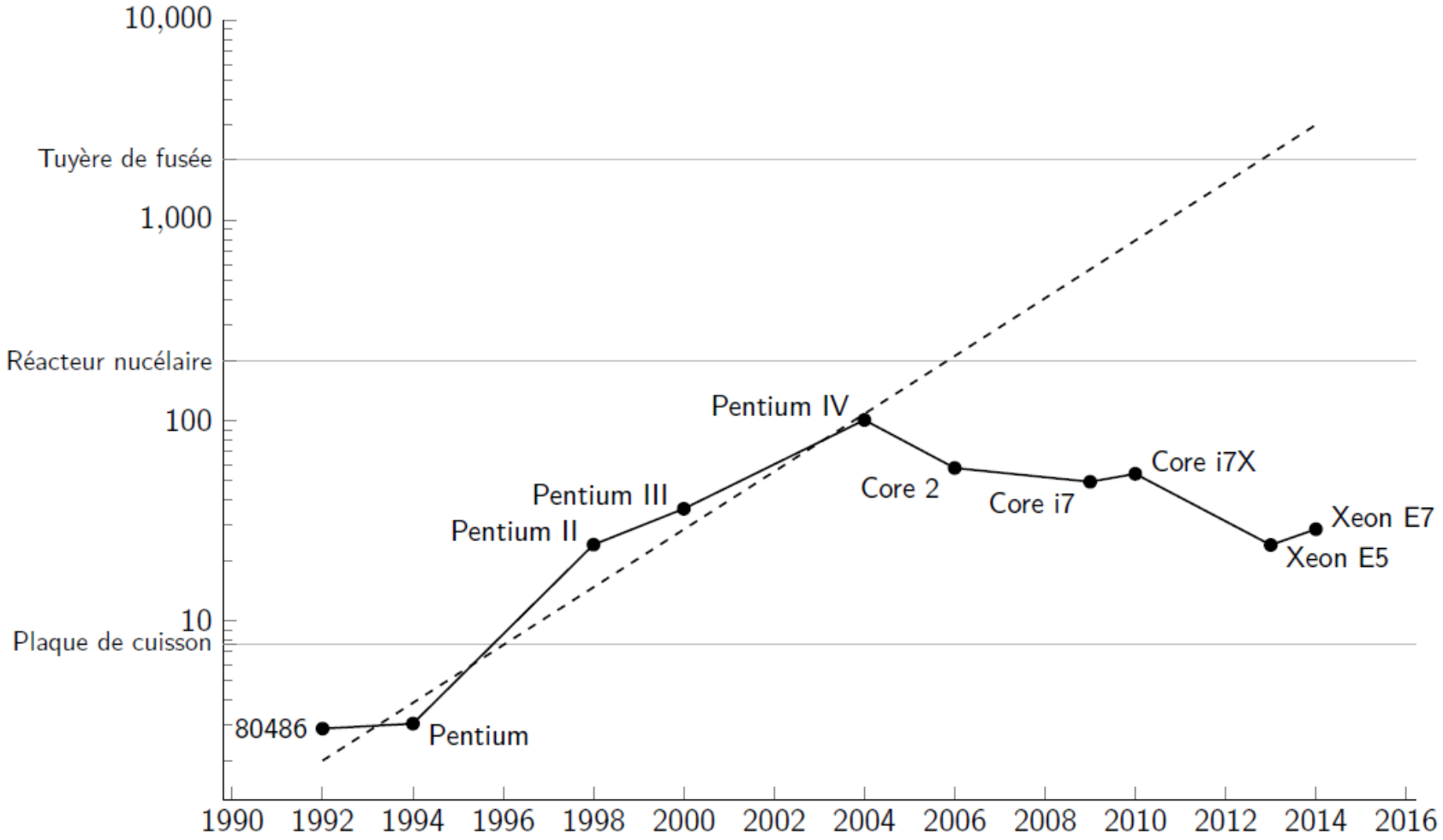
A la compensation de fréquence par parallélisme



Source: cours Intro Sys Emb ENSMA Henri Bauer

Fréquence et dissipation thermique

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

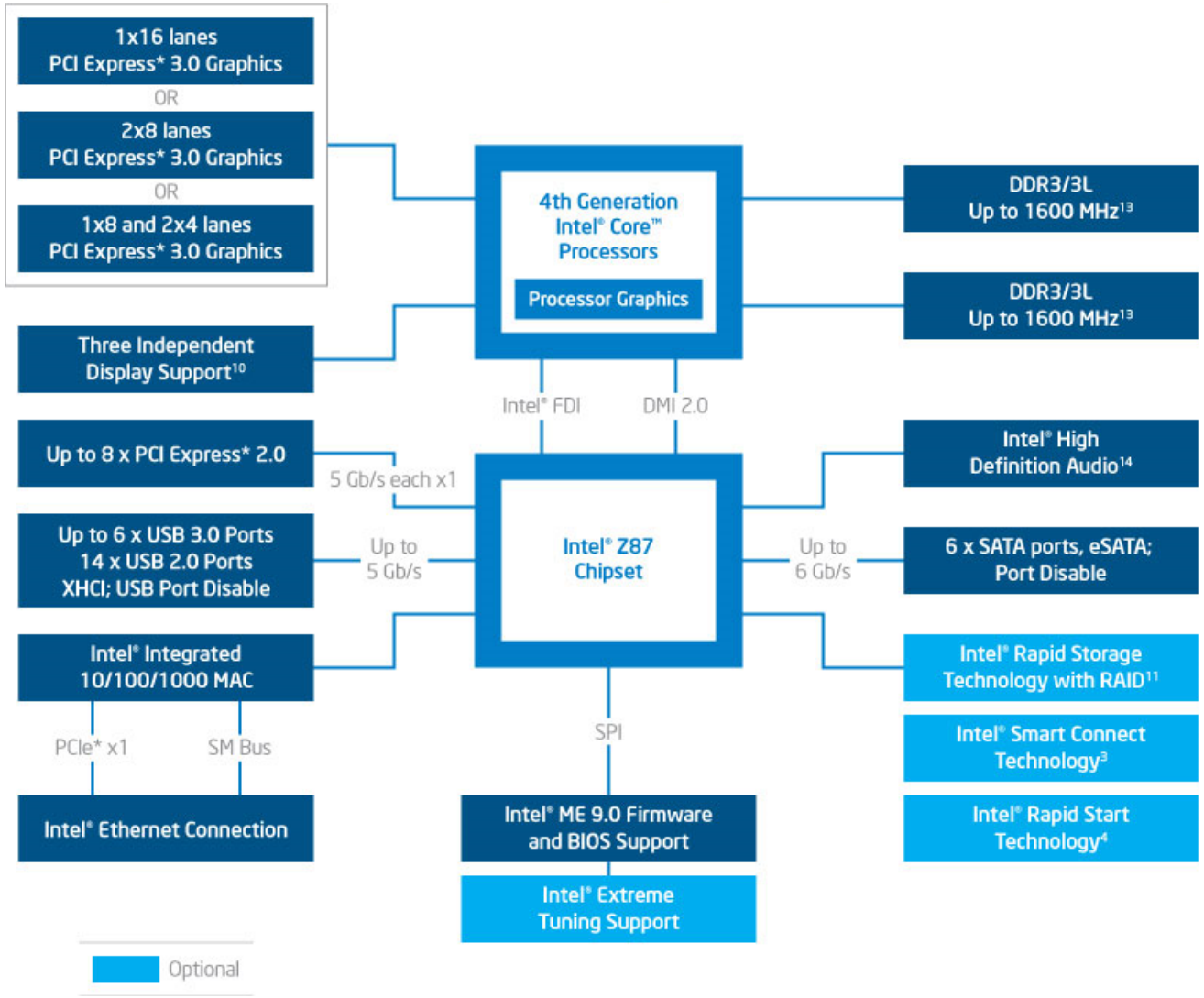


Source: cours Intro Sys Emb ENSMA
Henri Bauer

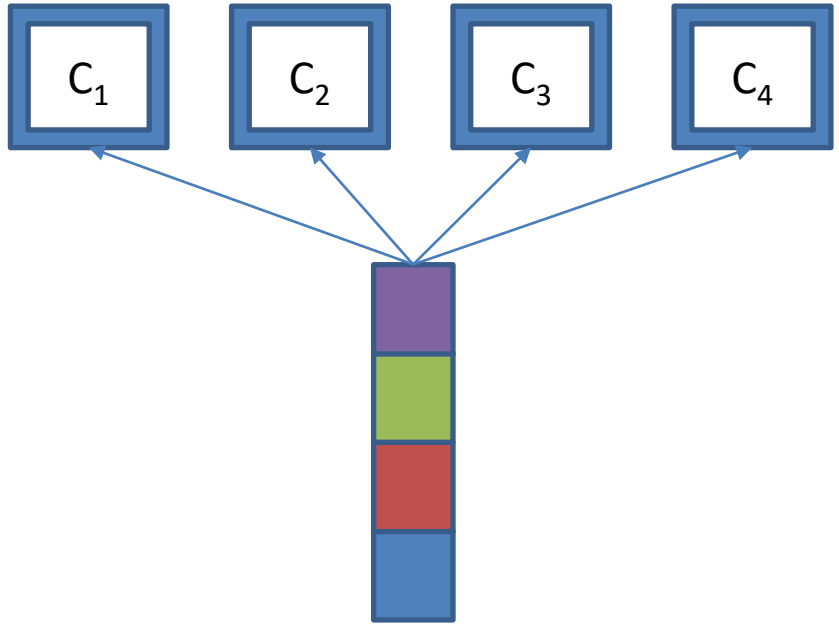
Hyperthreading

- Le code de la fonction pgcd pèse 48 octets, soit 12 accès mémoires en 32 bits.
- Lorsque les instructions 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
- Lorsque notre tâche exécutant pgcd doit charger d'autres données, les données chargées pour la seconde tâches peuvent être utilisées pour l'exécuter
- C'est le principe du processeur à 2 threads (hyperthreading, on parle aussi de SMT pour Simultaneous Multithreading)

Processeurs multicoeur



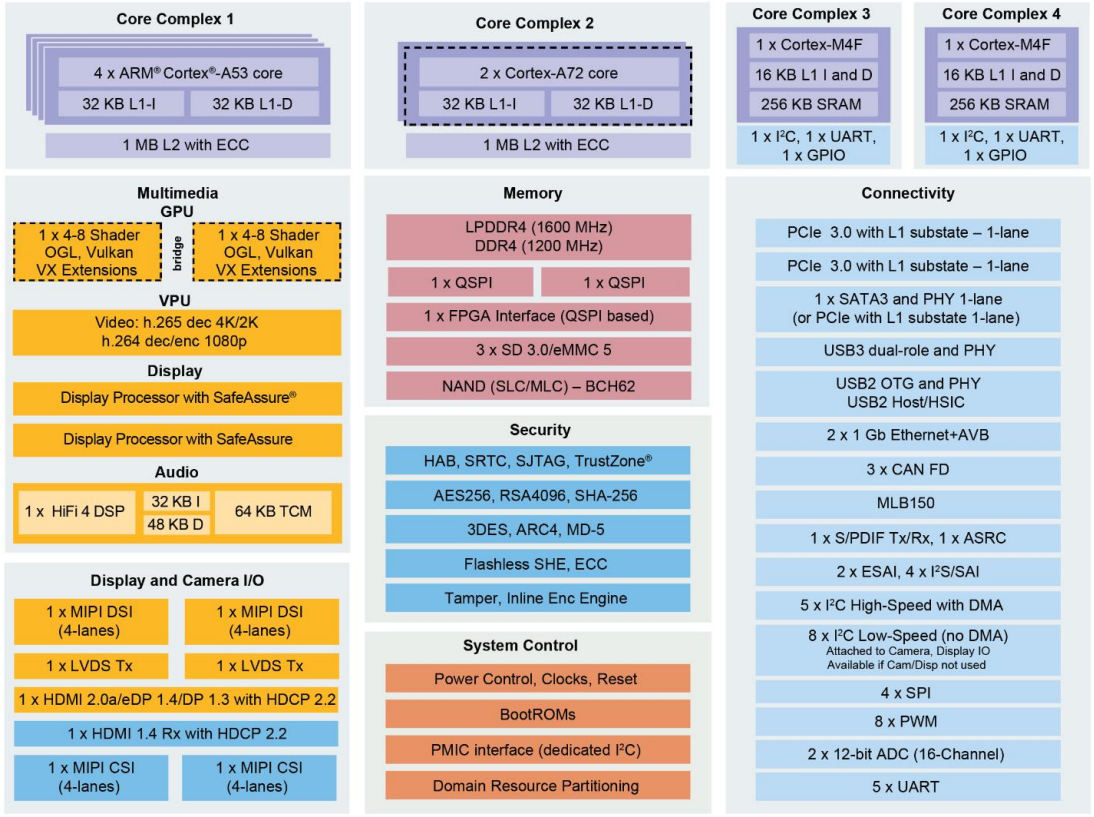
Modèle multicoeur ordonnancement global



File de tâches prêtes

Architecture MPSoC ARM

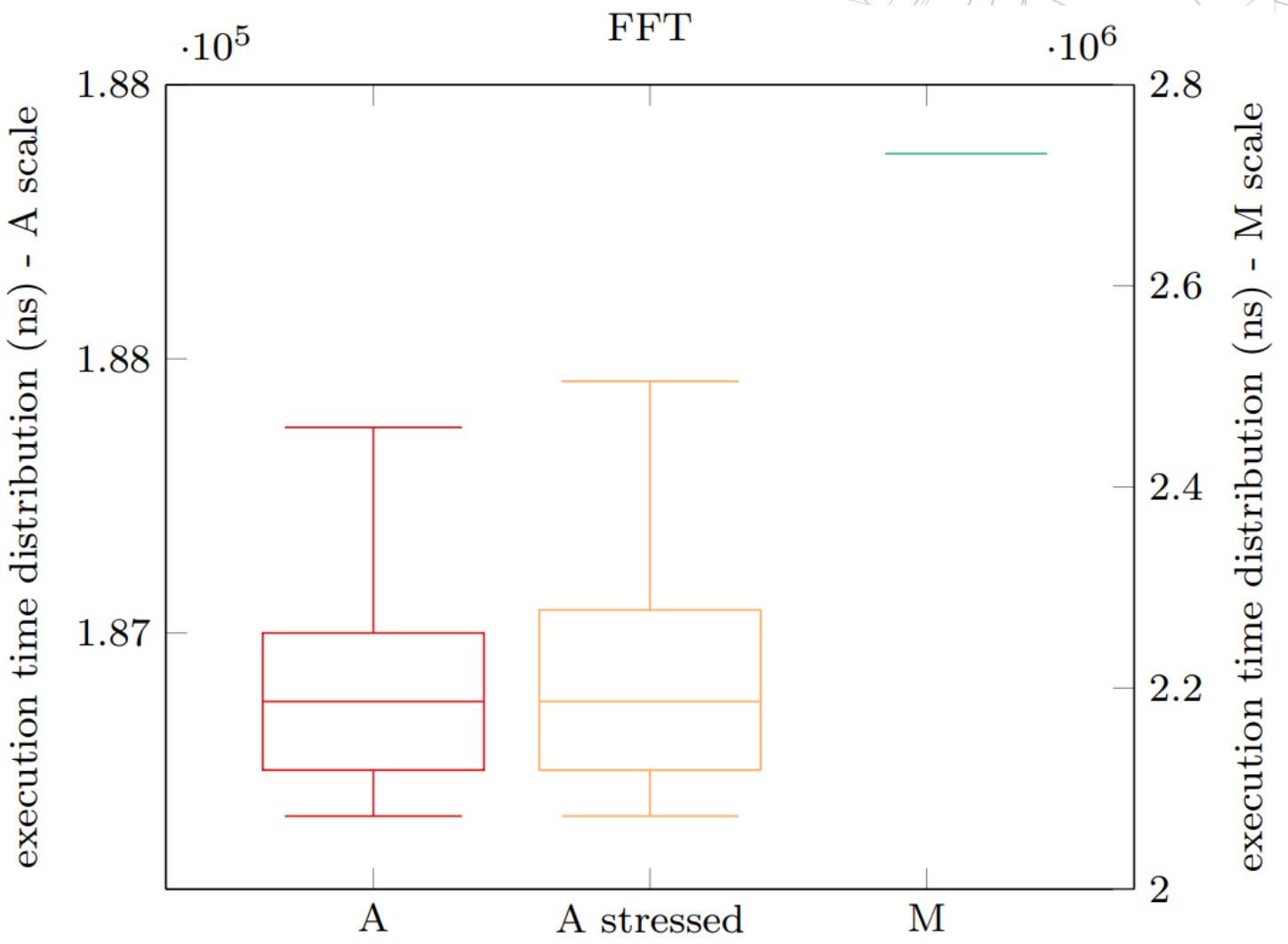
- Un MultiProcessor System on a Chip est une puce multicoeur intégrant plusieurs clusters de processeurs.



Available on certain product families Note: Accessing muxable controller's full capabilities is dependent upon board component choices.

Des comportements moins prédictibles

- Sur µProc Cortex A muni d'un FPU, exécution FFT prend de 187 à 188 µs
- Sur µC Cortex M (sans FPU), elle nécessite à peu près toujours 2,75 ms

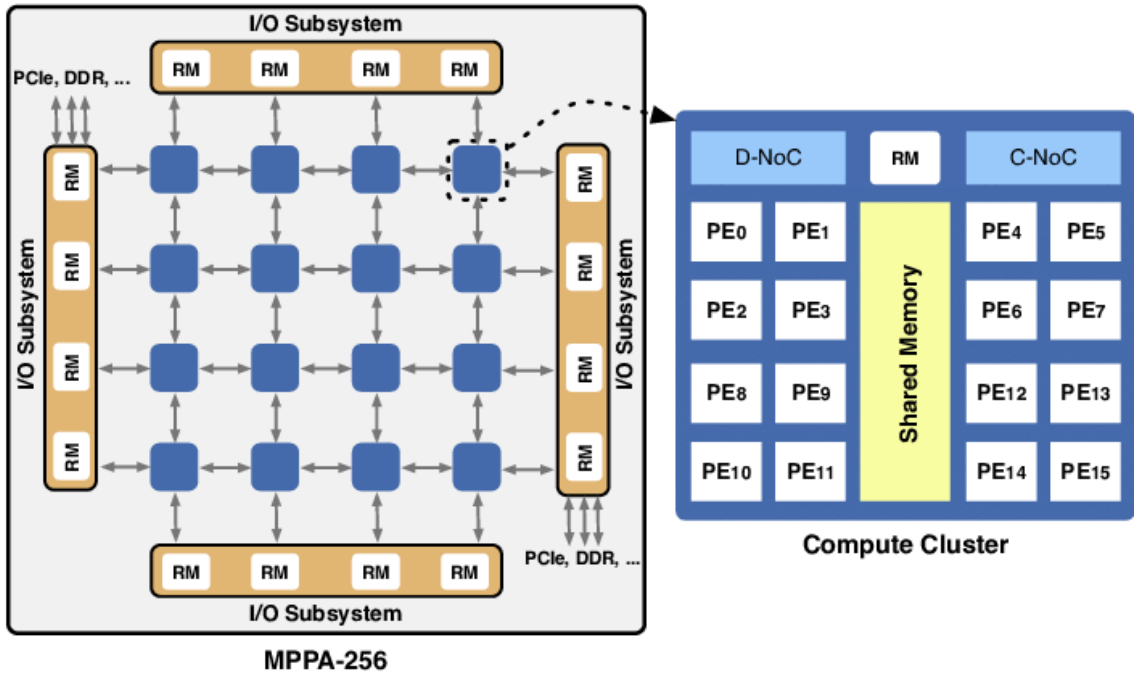


Contention multicoeur

- 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 ?

Mercredi après-m

Manycore



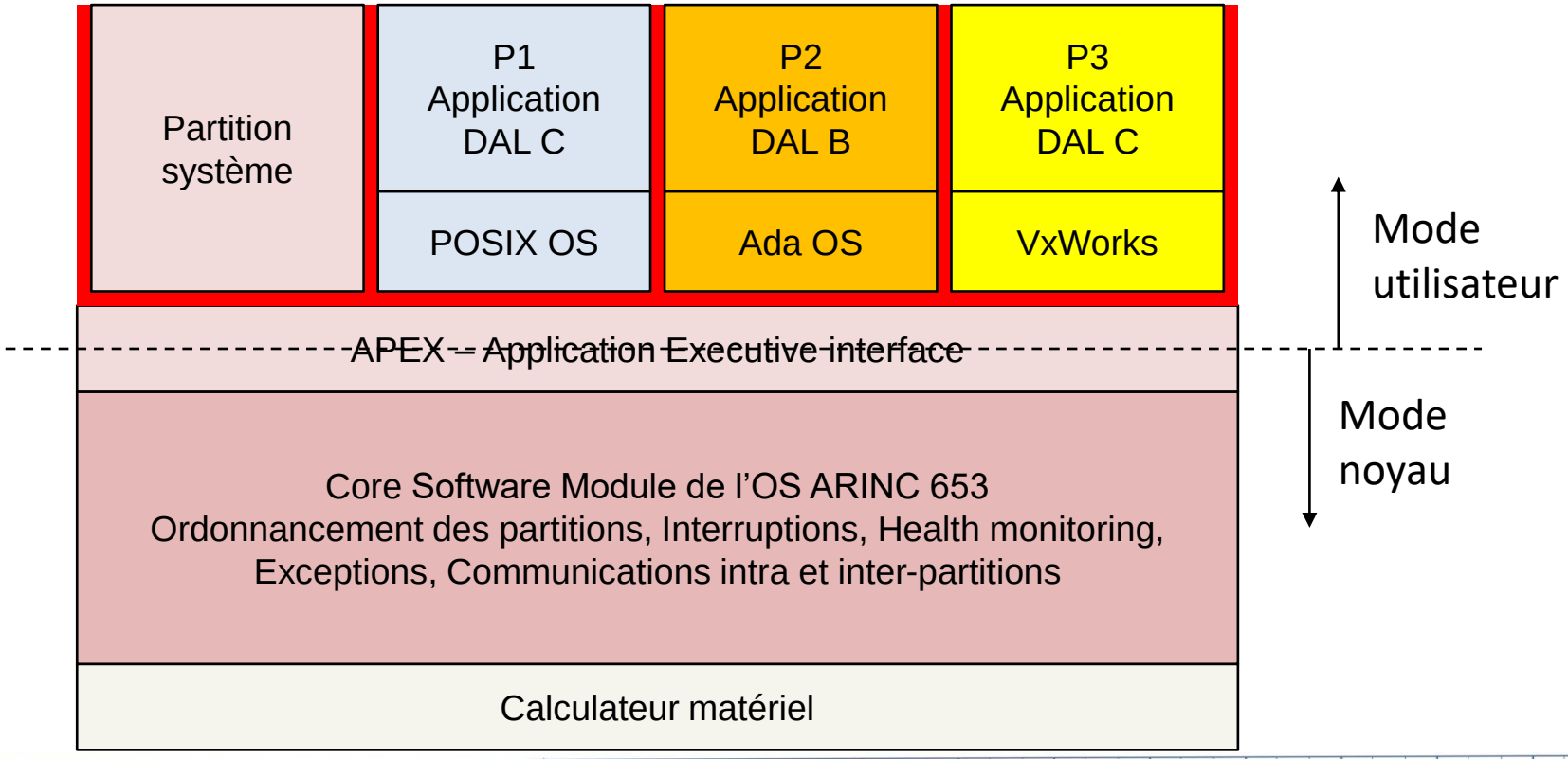
- Architecture complexe mais à très faible consommation
- Pourrait permettre de la dégradation gracieuse ?
- Moins sujette aux erreurs de micro architecture ?

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

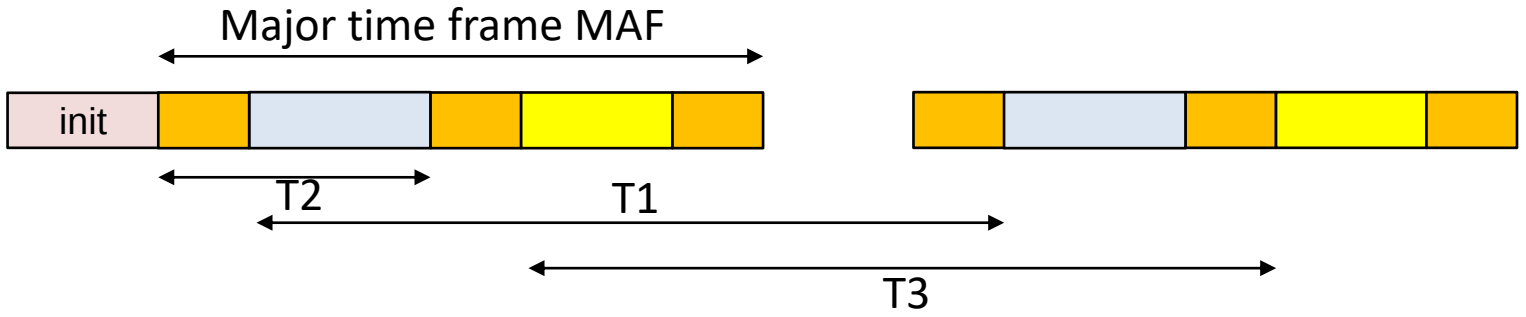
Le multicœur a-t-il sa place dans le critique ?

- CAST32A tente de définir des règles
- Utilisation dans l'avionique

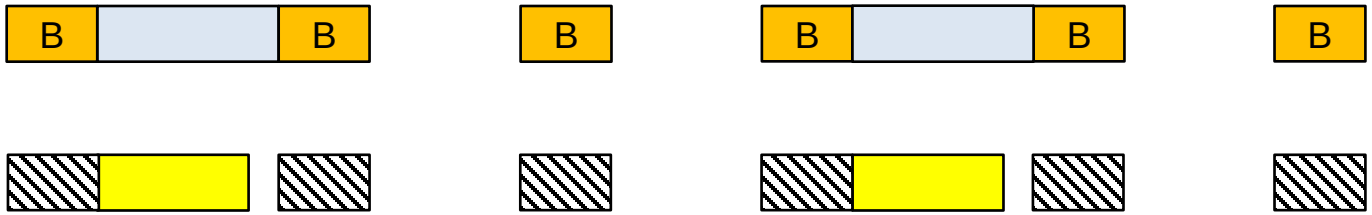


Premières utilisations du multicoeur

- Calculée statiquement



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

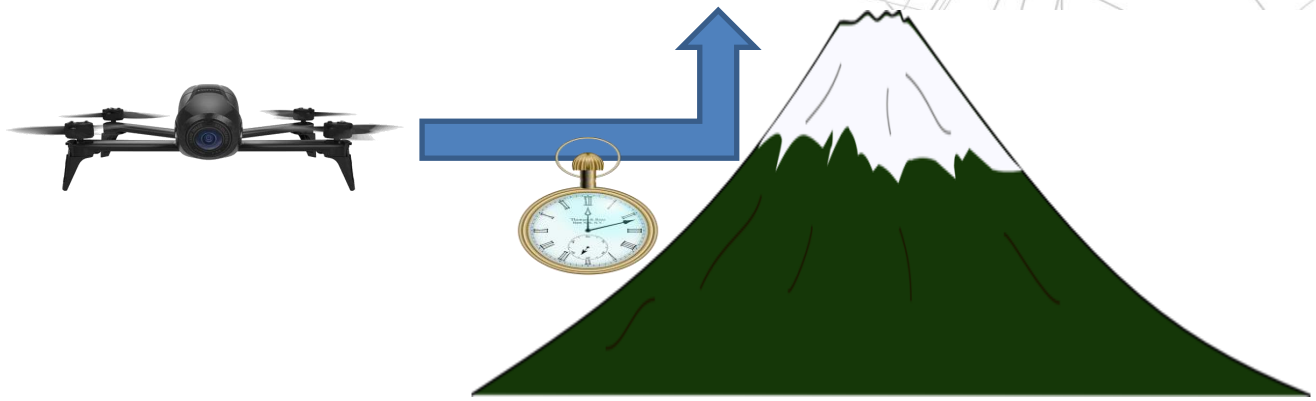


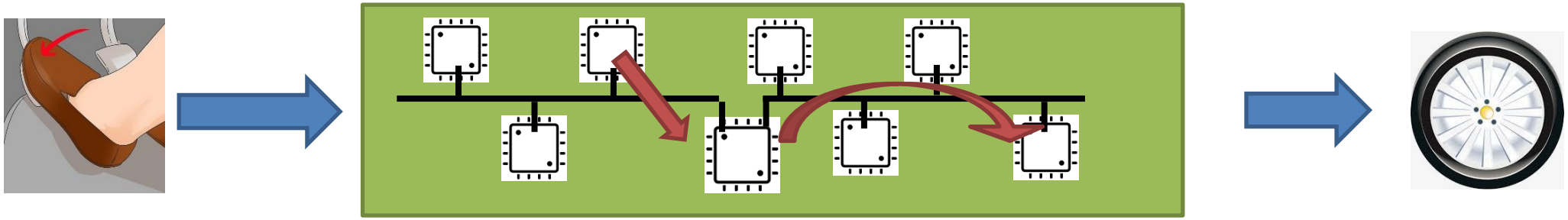
Mais d'où ça vient r, C, D, T, et ce modèle de Liu&Layland, ça représente quoi ?

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

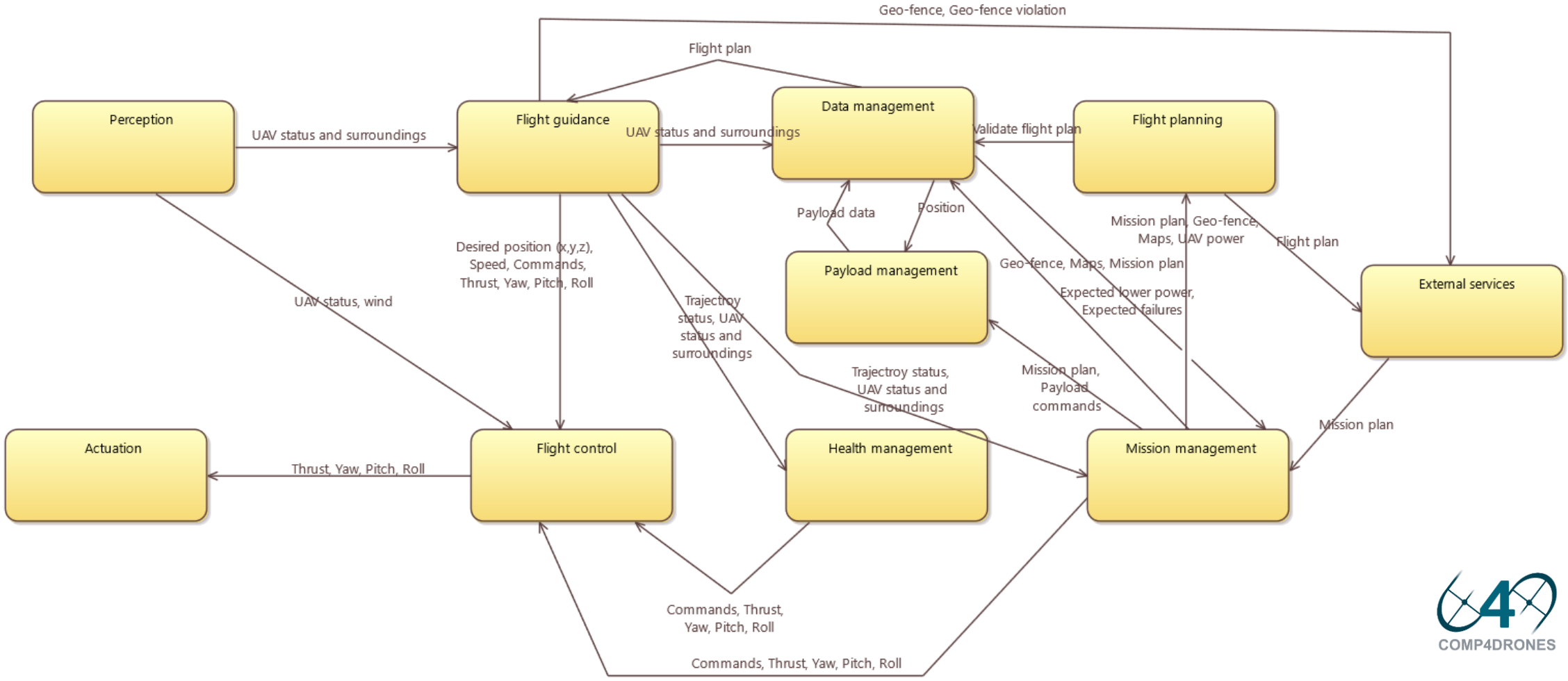




Conception de systèmes TR

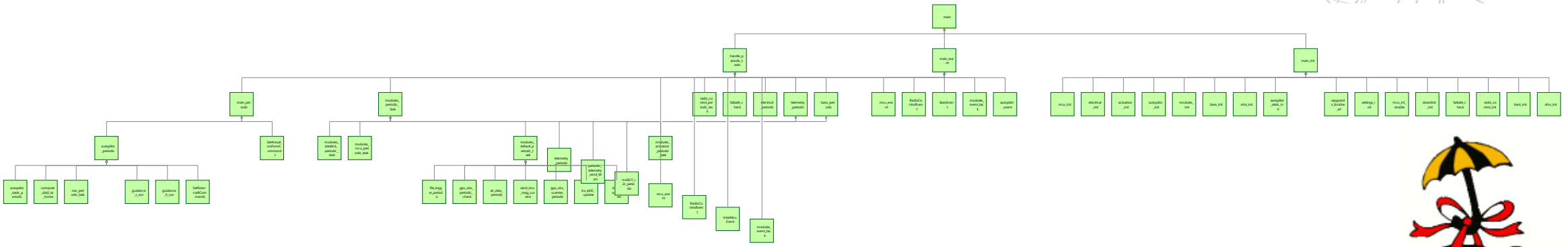
- Nombreux exemples de méthodes de conception
  - Arcadia, AUTOSAR, etc.
- Fonctions de haut niveau (~Use Case)
- Décomposition fonctionnelle (flots de données)
- Mapping sur architecture logicielle (threads, processus, OS, partition) et matérielle (calculateurs, réseaux)
- Nouvelles exigences apparaissent

Fonctions de haut niveau drone



Source: ECSEL H2020 Comp4Drones

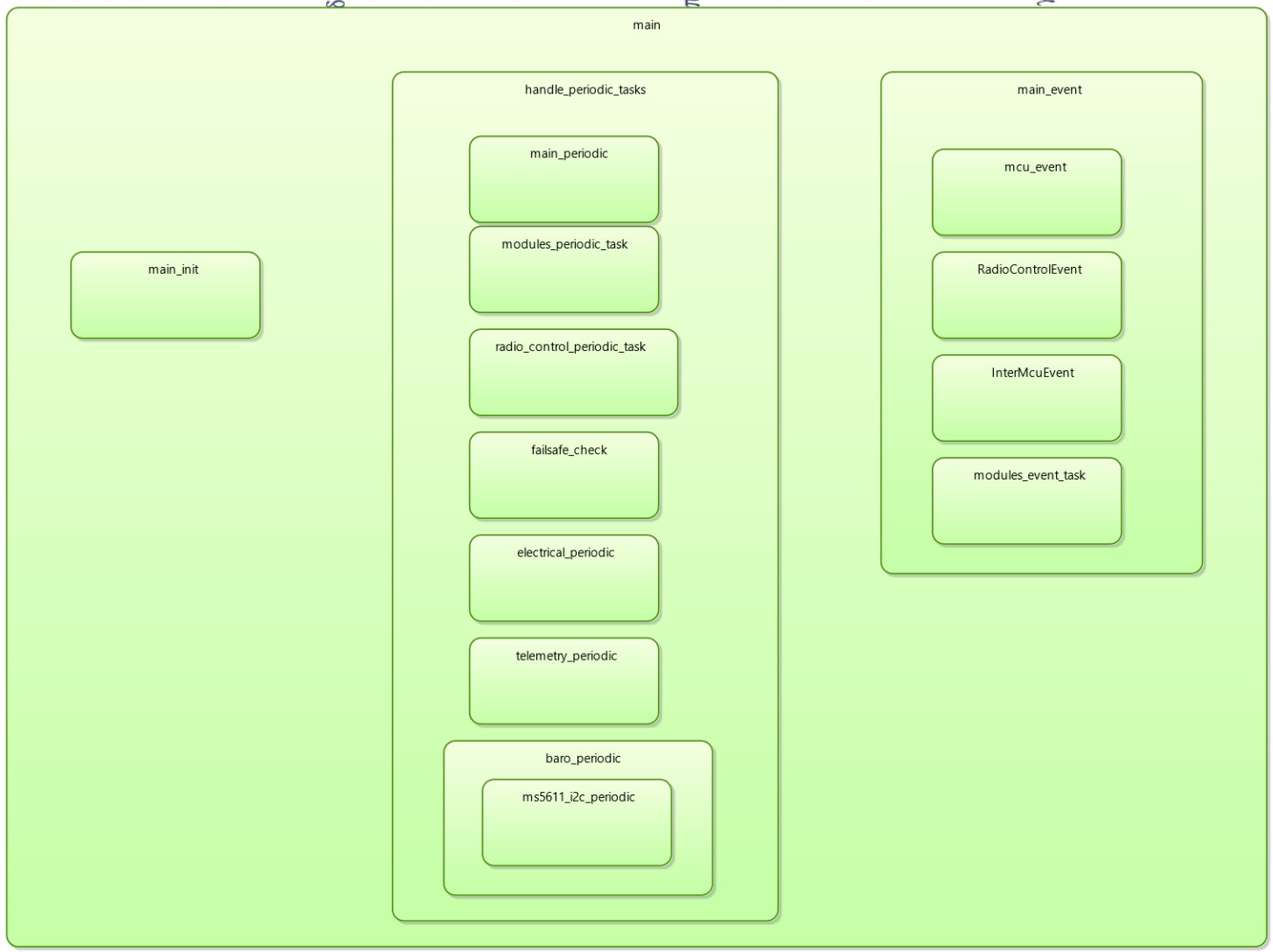
Autopilote Paparazzi



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

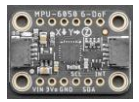


Exemple drone bebop 2

- Calculateur dual core Cortex A9 + GPU + 8GO mémoire flash
  - Linux SMP PREEMPT

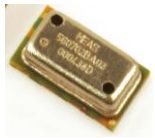
Jeudi matin

- 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

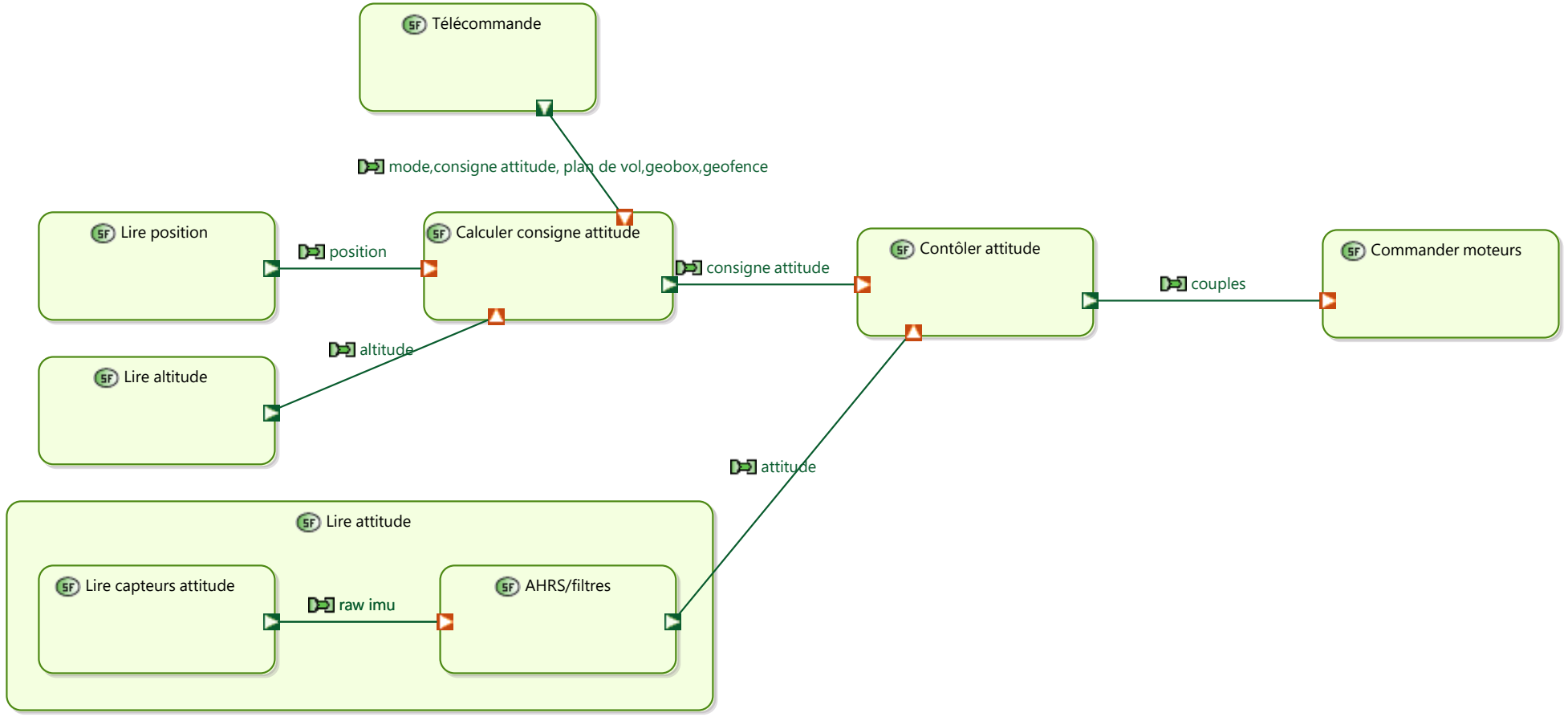
- Créent différentes contraintes (notamment temporelles)
- Autorisent ou non différents modes de réaction logicielle
  - Génération ou non d'interruption ⇒ scrutation ou événement
  - Mode push, pull, ou requête-réponse
    - ✓ Push : une ISR, voire une tâche peut être déclenchée qui peut déclencher 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

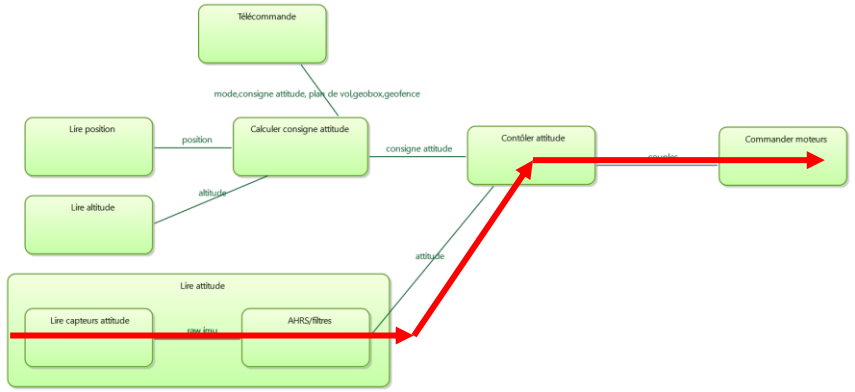
Archi logicielle et sûreté de fonctionnement

❑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

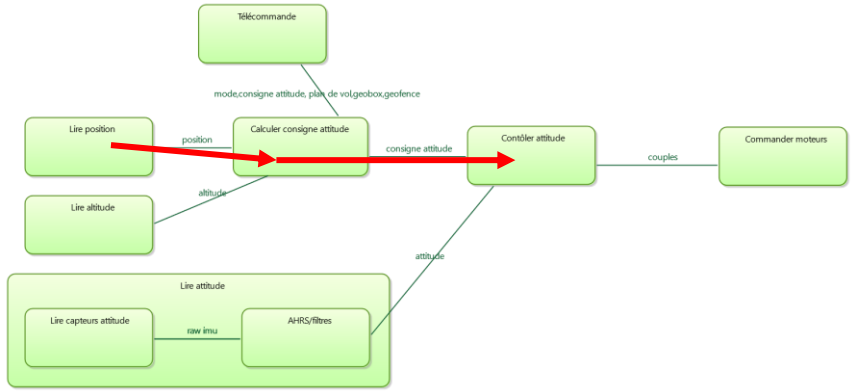
- Interroger et lire via I2C 3-accél, 3-gyros, température
  - 1 octet écrit, 14 octets lus
  - ~10 bits par octets à 400kHz (fréquence bus I2C) :
    - ✓ 150/400000 = 0,375 ms (15% de 2,5 ms)
- Interroger et lire via I2C 3-magneto
  - ~0,225 ms (9% de 2,5 ms)
- Filtrage et AHRS parfois gourmand
    - ✓ 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

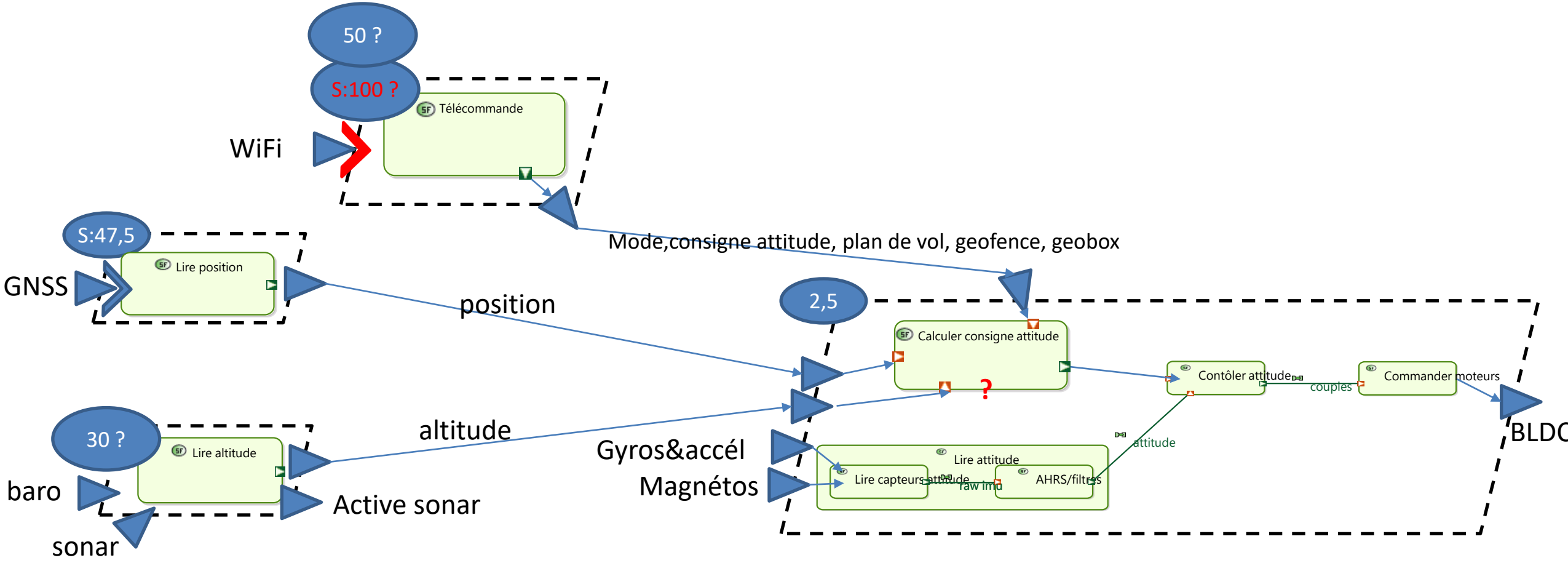
Chaîne fonctionnelle geofencing

- 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

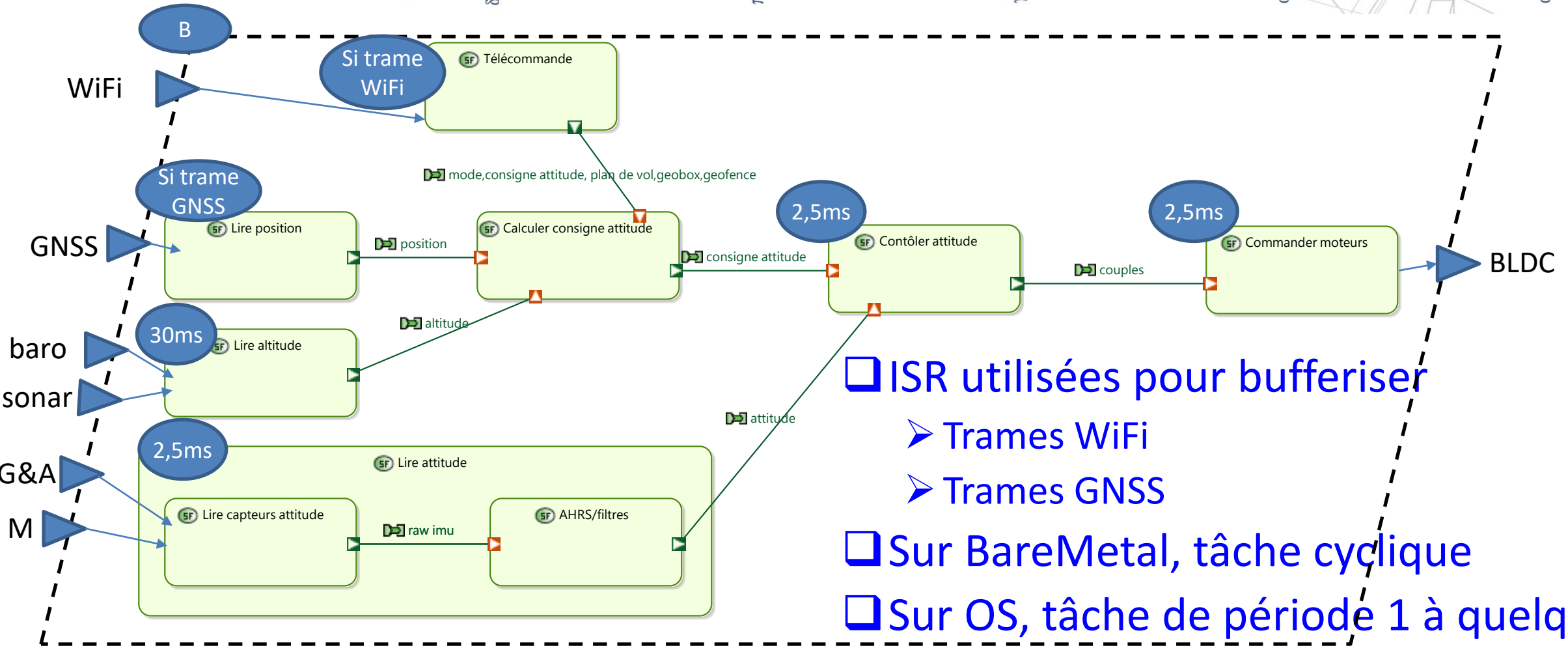
Contraintes de temps

- 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

Des systèmes souvent répartis

- Plupart des systèmes embarqués temps réel critiques sont répartis sur plusieurs plateformes reliées par un ou des réseaux
  - Permet aussi la redondance
    - ✓ Si calculateur victime d'au plus 1 panne pour 10^3 heures, 2 calculateurs en panne simultanément pour 10^6 heures
- Pas le temps d'en parler ici mais ETR donne une grande place aux réseaux cette année

jeudi après-m **vendredi matin**

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? 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

Merci, et bonne école d'été!!!