Institut
Mines-Télécom

Bus et réseaux

...ou comment bien communiquer

Alexis Polti

Licence de droits d'usage

Contexte académique } sans modification

Par le téléchargement ou la consultation de ce document, l'utilisateur accepte la licence d'utilisation qui y est attachée, telle que détaillée dans les dispositions suivantes, et s'engage à la respecter intégralement.

La licence confère à l'utilisateur un droit d'usage sur le document consulté ou téléchargé, totalement ou en partie, dans les conditions définies ci-après, et à l'exclusion de toute utilisation commerciale.

Le droit d'usage défini par la licence autorise un usage dans un cadre académique, par un utilisateur donnant des cours dans un établissement d'enseignement secondaire ou supérieur et à l'exclusion expresse des formations commerciales et notamment de formation continue. Ce droit comprend :

- le droit de reproduire tout ou partie du document sur support informatique ou papier,
- le droit de diffuser tout ou partie du document à destination des élèves ou étudiants.

Aucune modification du document dans son contenu, sa forme ou sa présentation n'est autorisé.

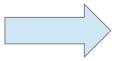
Les mentions relatives à la source du document et/ou à son auteur doivent être conservées dans leur intégralité.

Le droit d'usage défini par la licence est personnel, non exclusif et non transmissible.

Tout autre usage que ceux prévus par la licence est soumis à autorisation préalable et expresse de l'auteur :

alexis.polti@telecom-paristech.fr

Plan



- Définition
- Structure générique
- Topologies
- Protocoles
- Arbitrage
- Timings
- Augmentation des performances
- Bus embarqués usuels

Bus

- **Définition canonique**
  - Dispositif non bouclé reliant plusieurs composants, sous-ensembles ou matériels pour permettre entre eux l'apport d'énergie **et** la circulation d'informations.

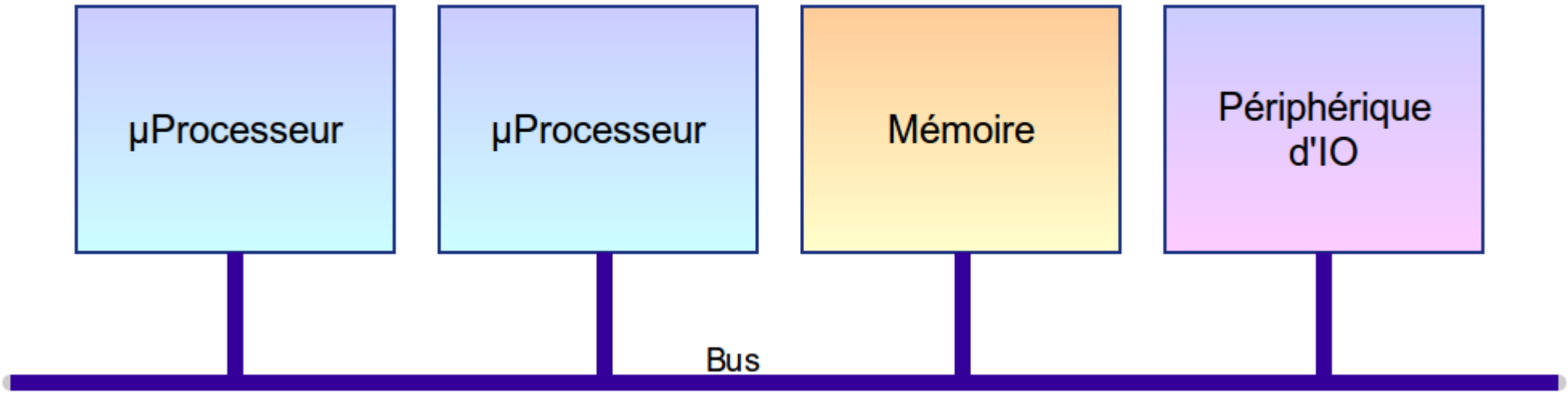
  - C'est un type particulier de réseau d'interconnexion
    - le plus simple
    - le plus répandu

- **Définition usuelle**
  - Moyen de communication partagé

Bus

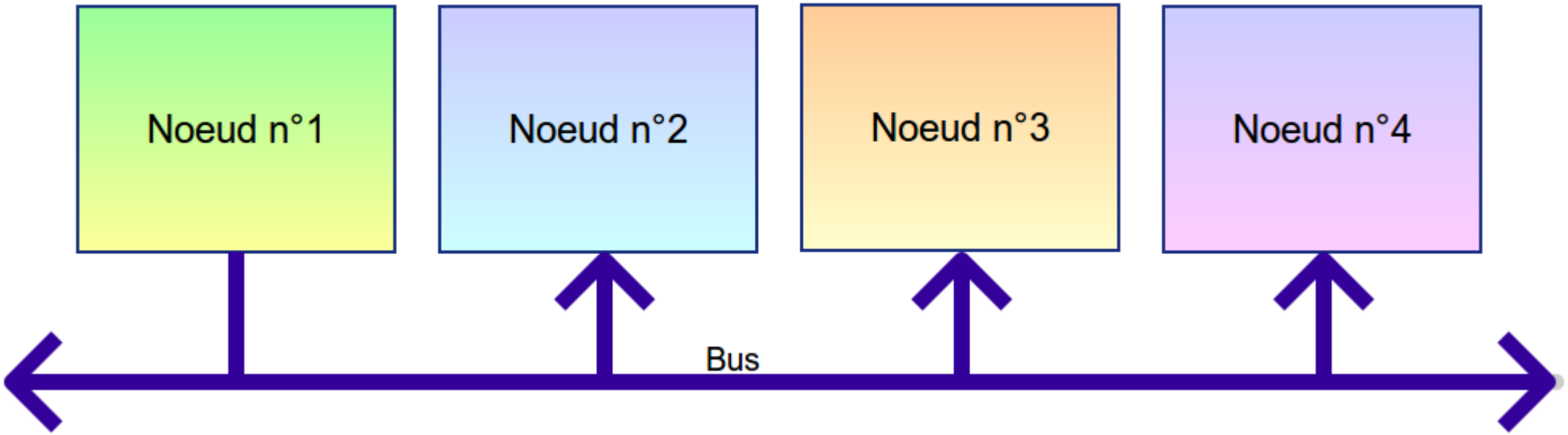
- Nœuds interconnectés par un médium de communication partagé

- Comprend :
  - un médium de communication, physique ou virtuel
  - un protocole de communication



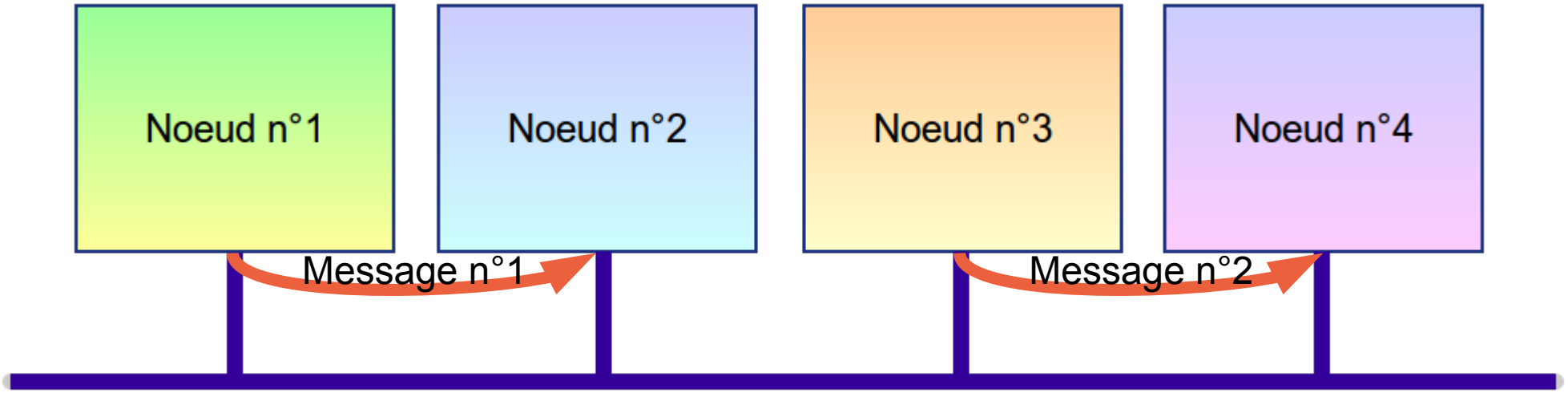
Bus vs. réseaux d'interconnexion

- Propriétés des bus
  - broadcast: possibilité de joindre plusieurs destinataires en même temps sans surcoût
  - distribution d'information globale
  - peut être remplacé par une garantie d'atomicité



Bus vs. réseaux d'interconnexion

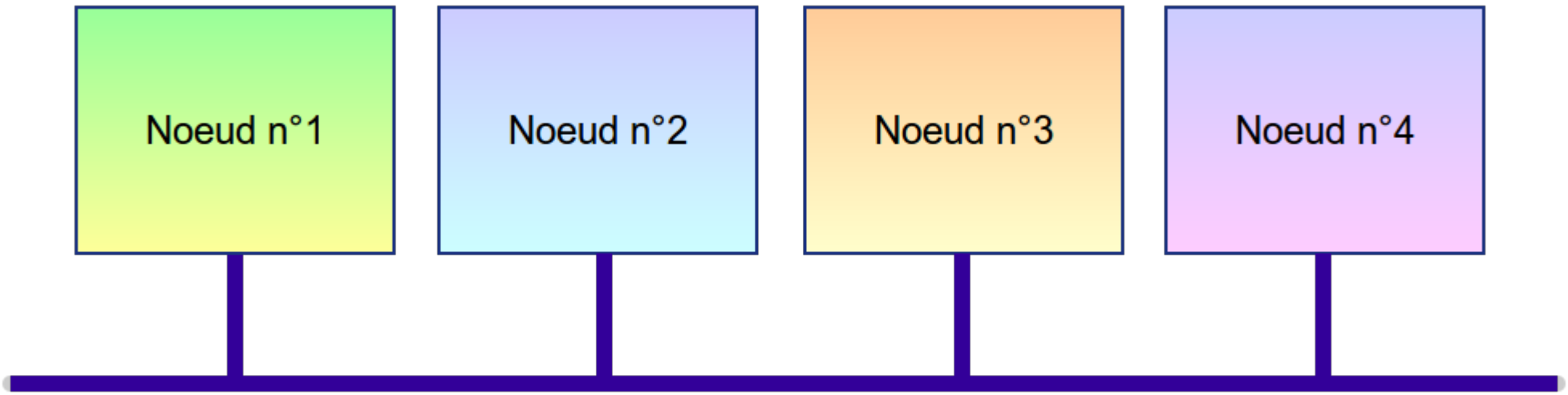
- Propriétés des bus
  - sérialisation:
    - un seul message est émis à un instant donné
    - les messages sont identifiés sans ambiguïté
  - messages totalement ordonnés



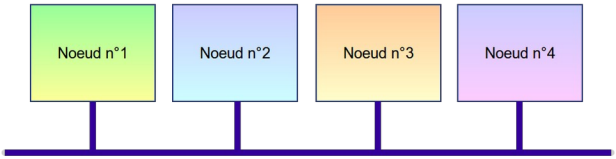
Bus vs. réseaux d'interconnexion

- Propriétés des bus
  - sérialisation et broadcast permettent d'assurer la cohérence des échanges et d'un système (cache snooping)

- En pratique, ces propriétés ne sont pas forcément assurées.



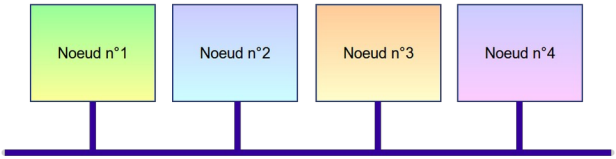
Bus



- Outil fondamental de construction de systèmes larges et complexes

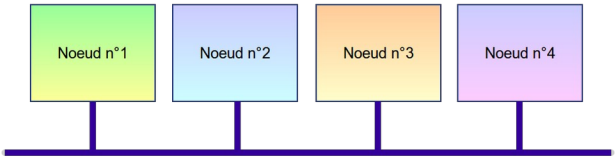
- Un bus électrique est défini par :
  - conventions mécaniques et physiques
  - conventions électriques
  - conventions temporelles (timing)
  - conventions d'accès (protocole)

Performances des bus



- **Caractéristiques recherchées**
  - performances
    - bande passante
    - débit
    - latence
    - garantie de comportement (temps-réel)
  - efficacité
    - coût
    - dissipation
  - robustesse
    - tolérance aux fautes
    - facilité de maintenance, diagnosticabilité
    - sécurité, sûreté

Performances des bus

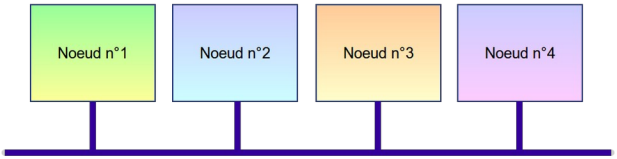


- **Mesure des performances**
  - Latence :
    - temps de réponse
    - temps total pour exécuter une transaction
  - Débit maximum :
    - mesure de la quantité d'information transmise avec succès dans un intervalle de temps
    - différent de la bande passante
    - = largeur des transactions * nombre moyen de transactions réussies/s

Plan

- Définition
- Structure physique
  - caractéristiques électriques
  - types usuels
- Topologies
- Protocoles
- Arbitrage
- Timings
- Augmentation des performances
- Bus embarqués usuels

Structure physique



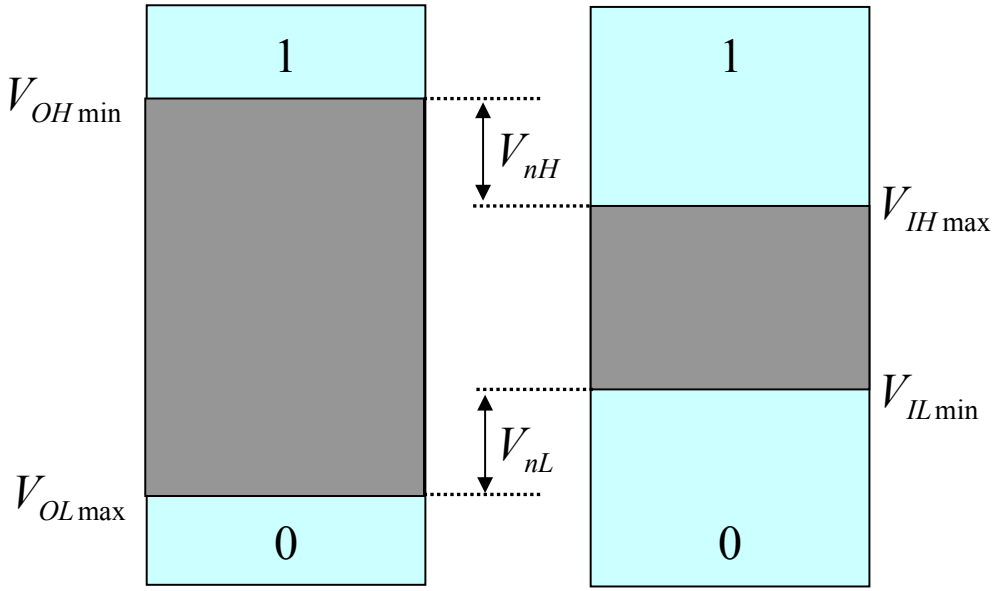
- **Usuellement :**
  - adresses
    - identifie la source ou la destination des données
  - données
    - contenu du message à transmettre
  - signaux de contrôle
    - qualification du message et informations de timing
    - exemple : read / write, horloge, ack, bs, ...
  - alimentation

Caractéristiques électriques

- Portes logiques
  - marge de bruit état haut : V_{nH}
  - marge de bruit état bas : V_{nL}
  - marge de bruit : $\min(V_{nL}, V_{nH})$

- typique : 0.5 – 1V

- potentiel de référence :
  - CMOS : moyenne pondérée de VCC et GND
  - TTL : offset de GND
  - ECL : offset de VCC



Caractéristiques électriques

- Signalisation absolue (single ended)
  - on affiche

  - on veut dire

  - il y a un chemin de retour, généralement implicite !

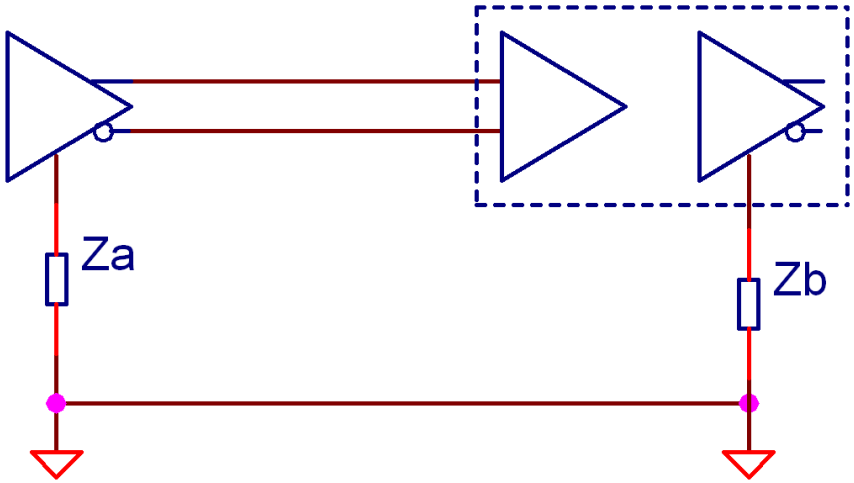
Caractéristiques électriques

- Signalisation absolue (single ended)
  - Le récepteur voit Vin – V_{GND}
  - sources d'erreurs :
    - Vin bruité
    - V_{GND} bruité

  - le problème vient du fait que GND sert à trop de choses :
    - référence de tension
    - chemin de retour du signal entrant
    - alimentation (chemin de retour d'un éventuel driver dans le même package que le récepteur)
  - principe de la signalisation double : séparer les rôles
    - un fil pour la référence (et chemin de retour)
    - un fil pour l'alimentation

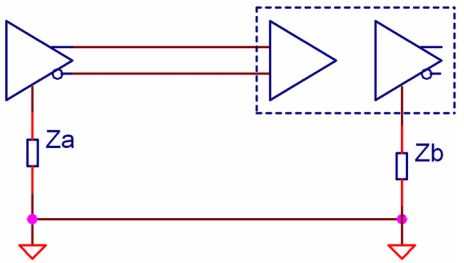
Caractéristiques électriques

- Signalisation double
  - transmission du signal ET de la référence associée



- avantages
  - immunité aux bruits sur GND
  - immunité au ground-bounce
  - immunité aux bruits couplés de façon identique dans les deux lignes
  - rayonne très peu

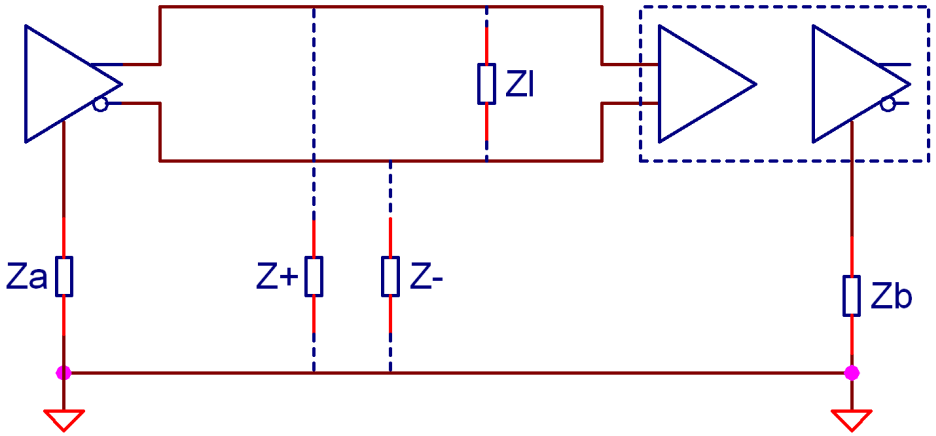
Caractéristiques électriques



- Signalisation différentielle
  - idéalement, $i_{GND} = 0$
    - sinon les deux signaux ne sont pas affectés de la même façon
  - en pratique
    - couplage de chaque ligne avec GND
    - ces couplages doivent être négligeables
      - soit extrêmement faibles
      - soit opposés, de façon à ce qu'ils se compensent
    - dans les deux cas, on aura toujours $i_{GND} \approx 0$
- Définition
  - *Signalisation différentielle* : transmission de V+ et V- avec :
    - dV- = -(dV+)
    - $i_{GND} \approx 0$
  - i_{GND} est appelé « *courant de mode commun* »

Caractéristiques électriques

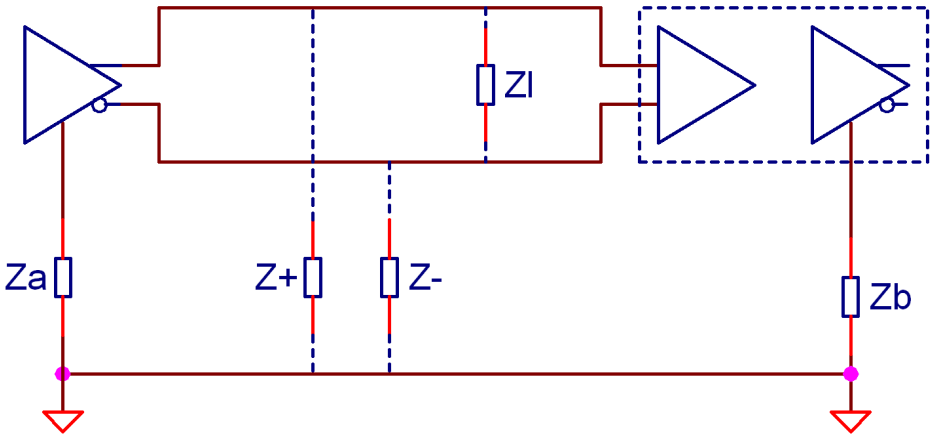
- Signalisation différentielle
  - modélisation :
    - couplage V+ / GND : Z+
    - couplage V- / GND : Z-
    - couplage entre V+ et V- : Zl



  - étude plus complexe qu'en single-ended
  - on veut $i_{GND} \approx 0$, donc $\frac{1}{Z_+} - \frac{1}{Z_-} \approx 0$

Caractéristiques électriques

- Signalisation différentielle
  - $i_{GND} \approx 0$: deux stratégies

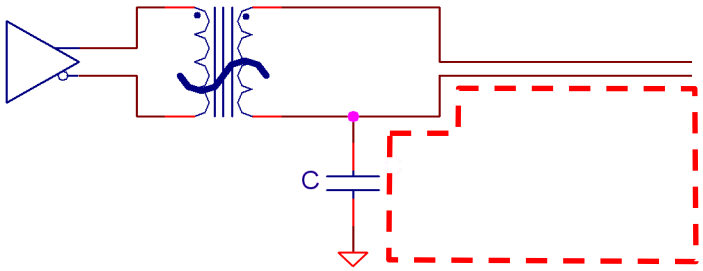


- couplage fort
  - exemple : paires torsadées
  - isolant épais : Z+ et Z- grands
  - torsadée : Z+ ≈ Z-
  - couplage :
    - entre brins : très fort (crosstalk 100%)
    - brins / masse : très faible

- balance précise
  - sur un PCB : impossible de rendre Z+ et Z- très grands
  - il faut donc les rendre précisément égaux :
    - exactement les mêmes dimensions pour les deux pistes
    - et driver correct…
  - pas besoin d'un couplage fort entre les deux pistes

Caractéristiques électriques

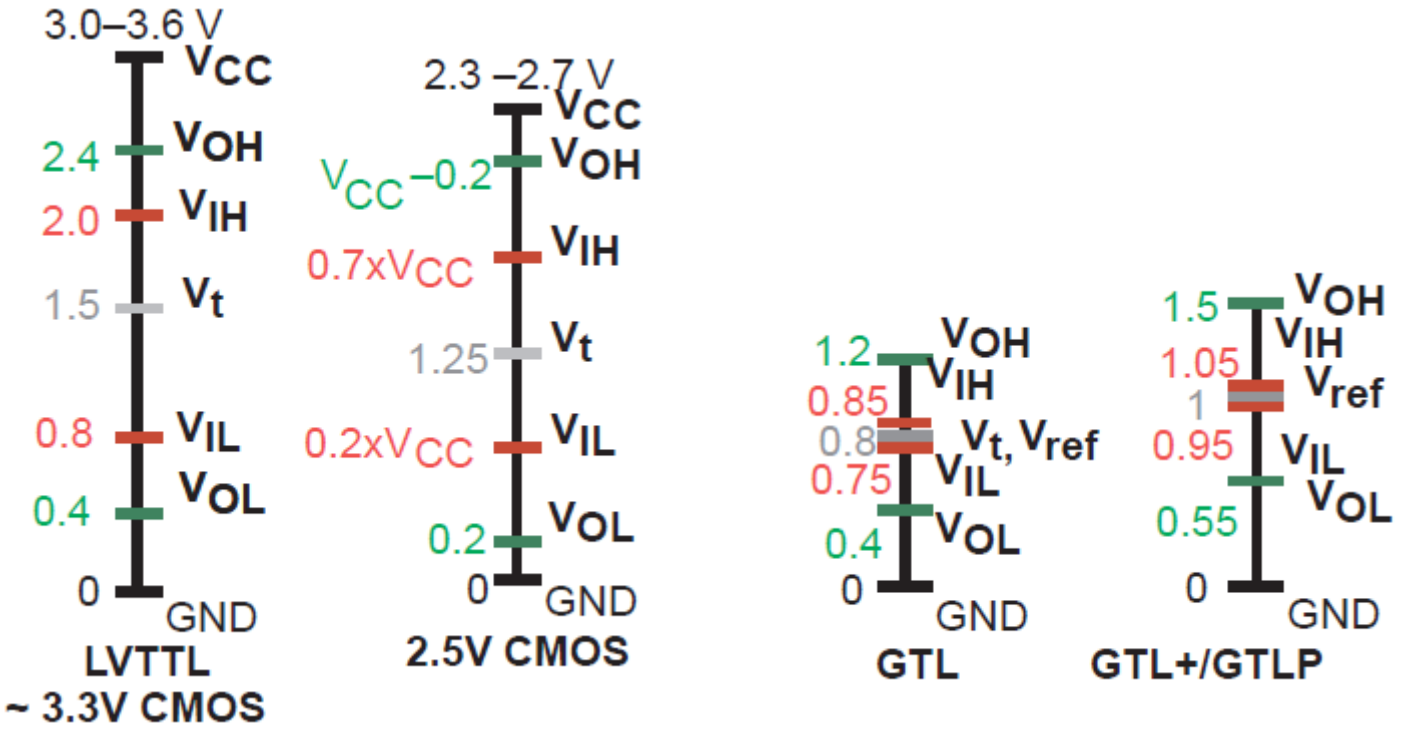
- Courant de mode commun
  - en pratique i_{GND} n'est pas nul : très problématique, surtout dans le cas de câbles
  - exemple : Ethernet 10BASE-T
    - C = 2pF, Tr=25ns, dV=2V
    - i = 160µA



  - la boucle est tellement large que i est largement suffisant pour empêcher la conformité CEM
  - conséquences :
    - minimiser à tout prix le courant de mode commun.
    - étude précise des impédances et des différents courants
    - cf. le cours d'intégrité du signal

Caractéristiques électriques

- **Niveaux de référence**
  - single ended

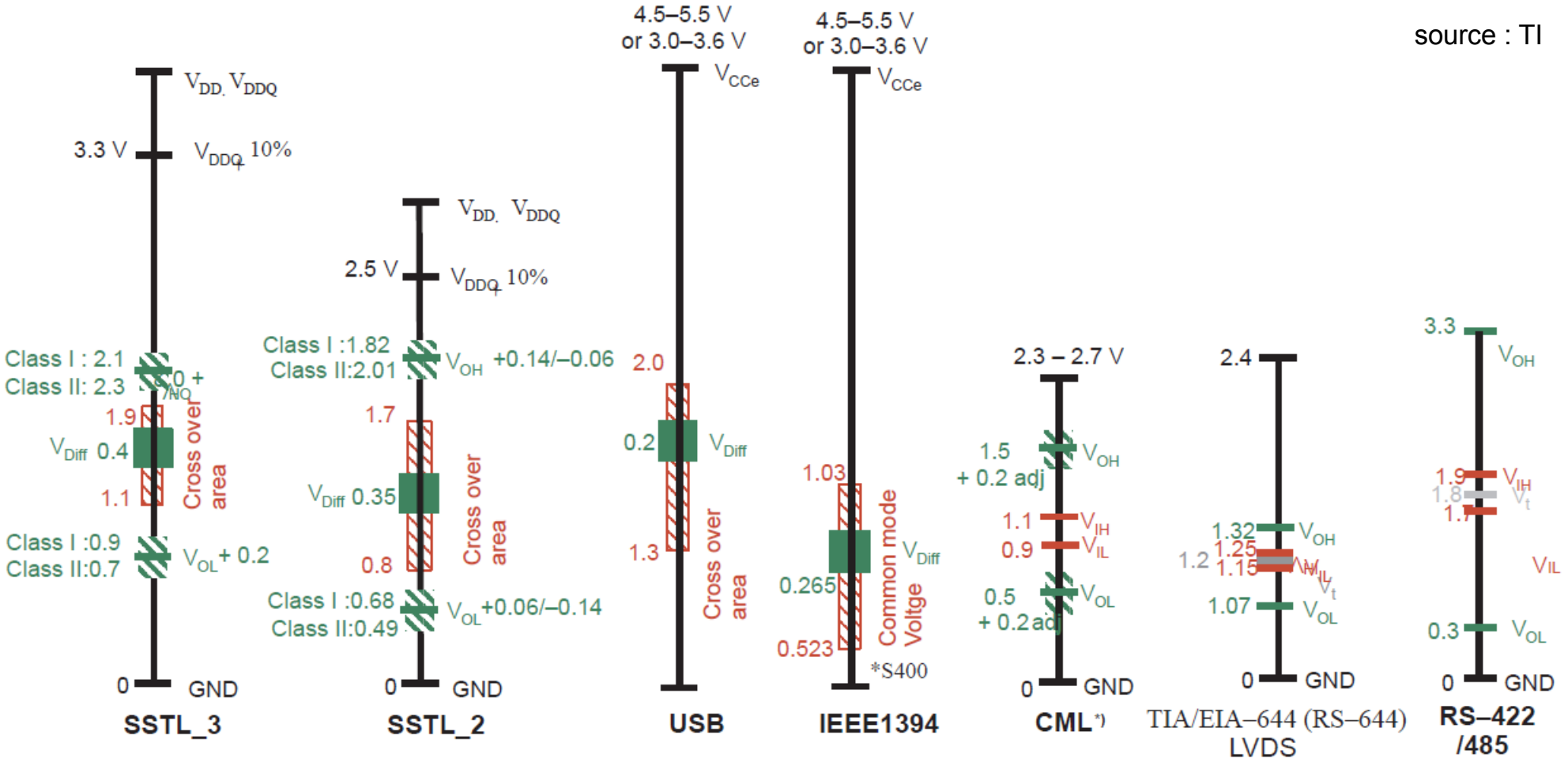


source : TI

Caractéristiques électriques

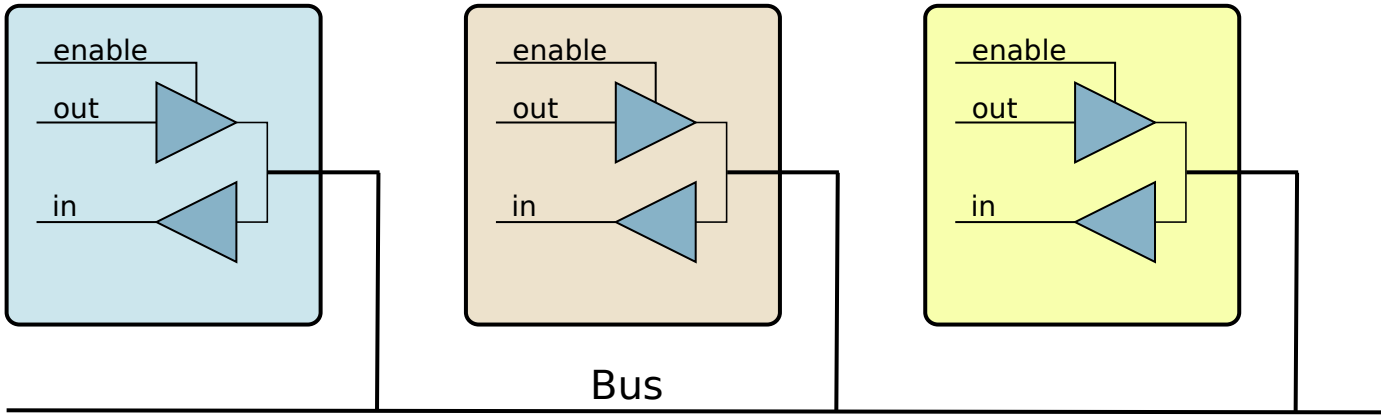
- **Niveaux de référence**
  - différentiel

source : TI



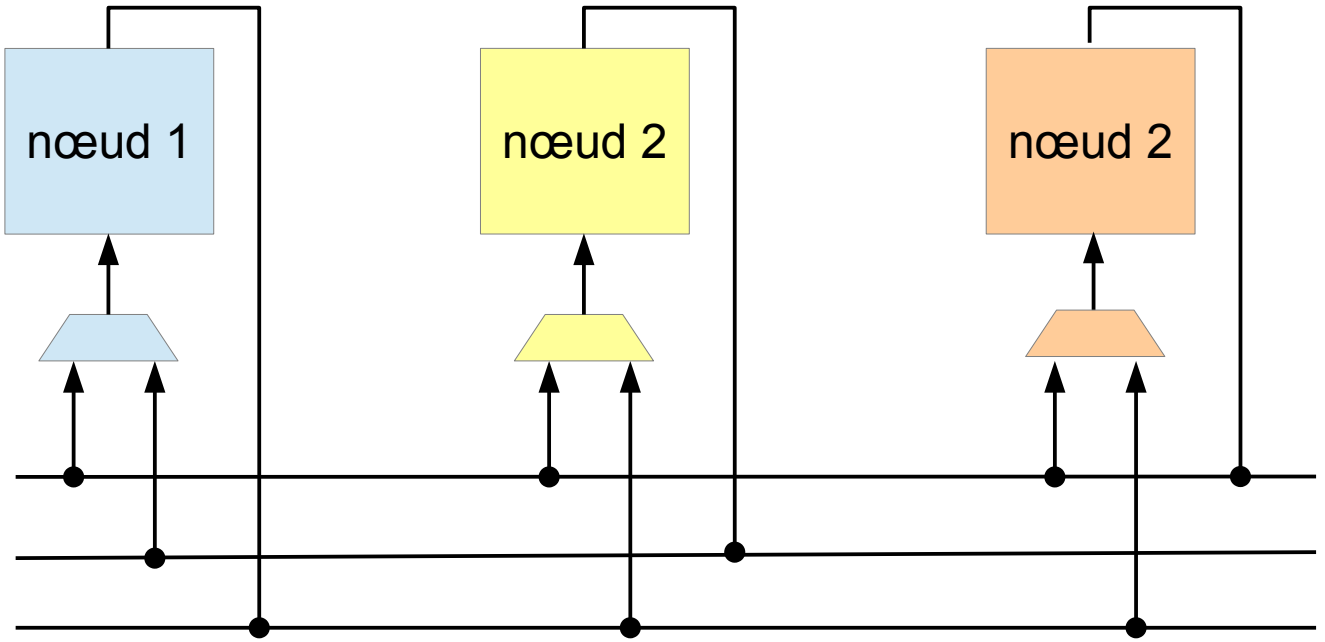
Caractéristiques électriques

- **Transactions bidirectionnelles**
  - drivers bidirectionnels
    - nécessite des drivers 3 états
    - contrôle précis des drivers pour éviter les court-circuits
    - timings complexes



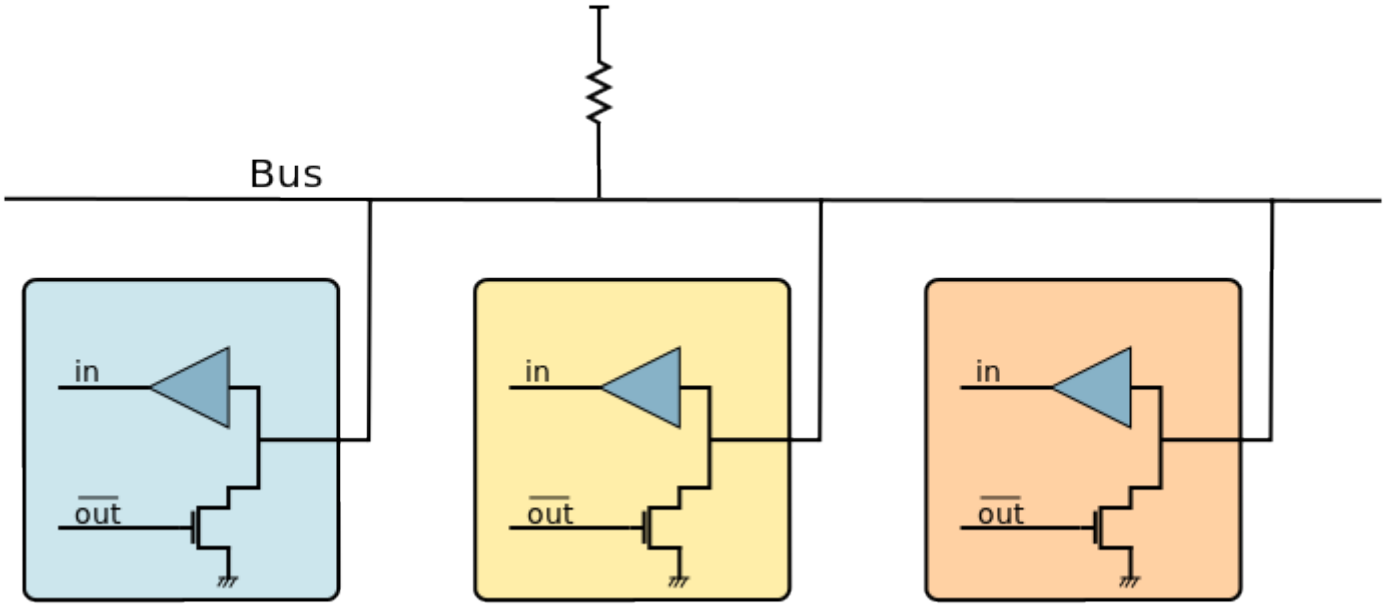
Caractéristiques électriques

- **Transactions bidirectionnelles**
  - drivers unidirectionnels
    - nécessite un multiplexage à l'arrivée
    - complexifie le câblage
    - facilite la gestion des timings



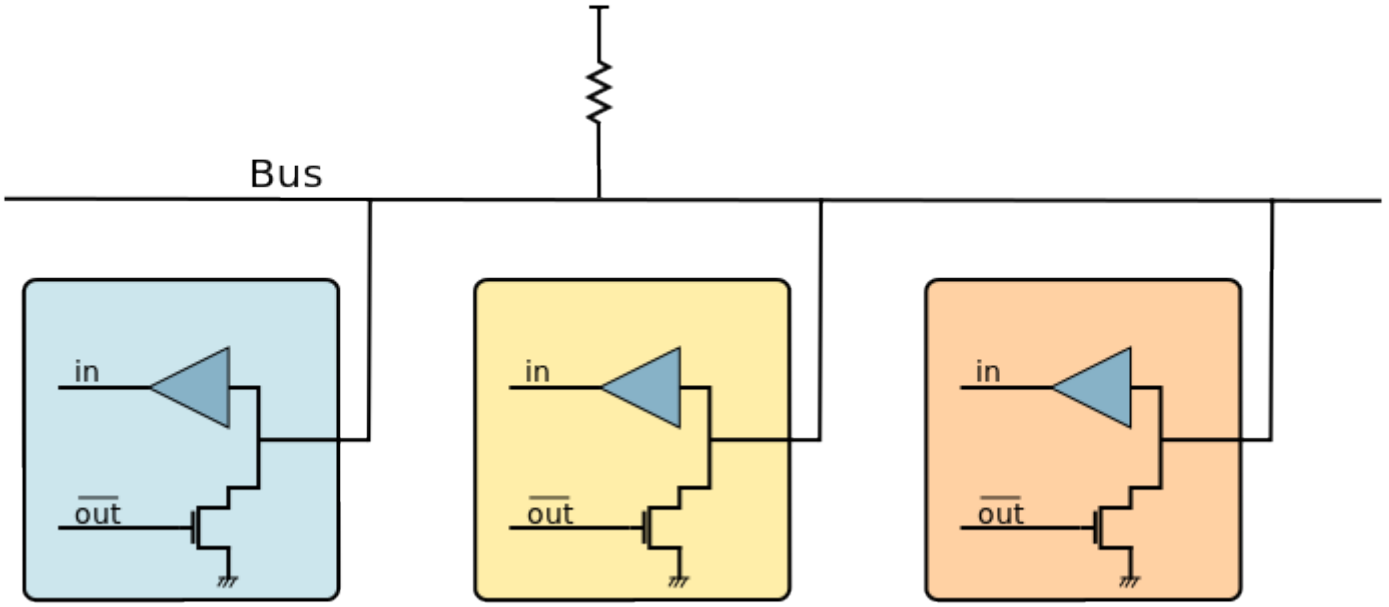
Caractéristiques électriques

- **Transactions bidirectionnelles**
  - drivers open drain (collecteur ouvert)
    - évite, par construction, les court-circuits
    - AND câblé
    - nécessite une résistance de rappel (pull-up)

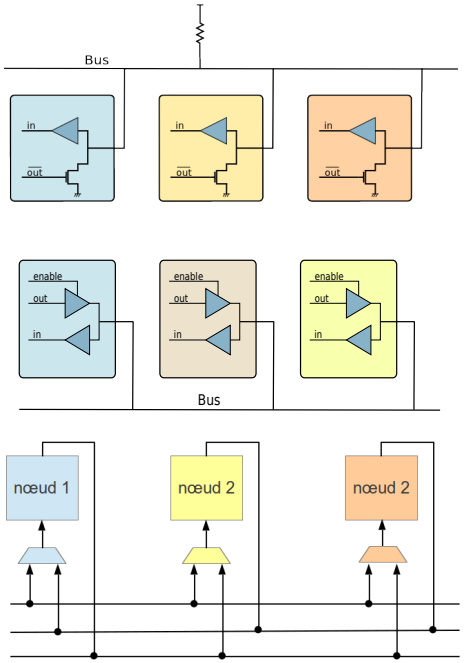


Caractéristiques électriques

- **Transactions bidirectionnelles**
  - drivers open drain (collecteur ouvert)
    - comment calculer la valeur de la résistance de rappel ?
      - courant maximal dans les drivers
      - fréquence minimale de fonctionnement

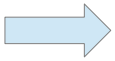


Caractéristiques électriques



- **Transactions bidirectionnelles**
  - charge du bus
    - due aux interconnexions
    - due aux capacités d'entrée
  - limite la fréquence maximale de fonctionnement du bus
  - comment augmenter cette fréquence ?
    - limiter l'excursion en tension
    - paires différentielles

Plan



- Définition
- Structure physique
  - caractéristiques électriques
  - types usuels / hiérarchie
- Topologies
- Protocoles
- Arbitrage
- Timings
- Augmentation des performances
- Étude de bus embarqués usu

Types de bus

- **Types de bus**
  - classification usuelle
    - local bus (bus processeur – mémoire)
    - backplane (fond de panier)
    - IO bus (bus d'entrée-sorties)
  - un système est souvent composé d'une hiérarchie de bus
    - nœud commun à deux bus : pont (bridge)
    - permet de :
      - limiter la charge des bus rapides
      - limiter les effets des latences avec périphériques lents

Types de bus

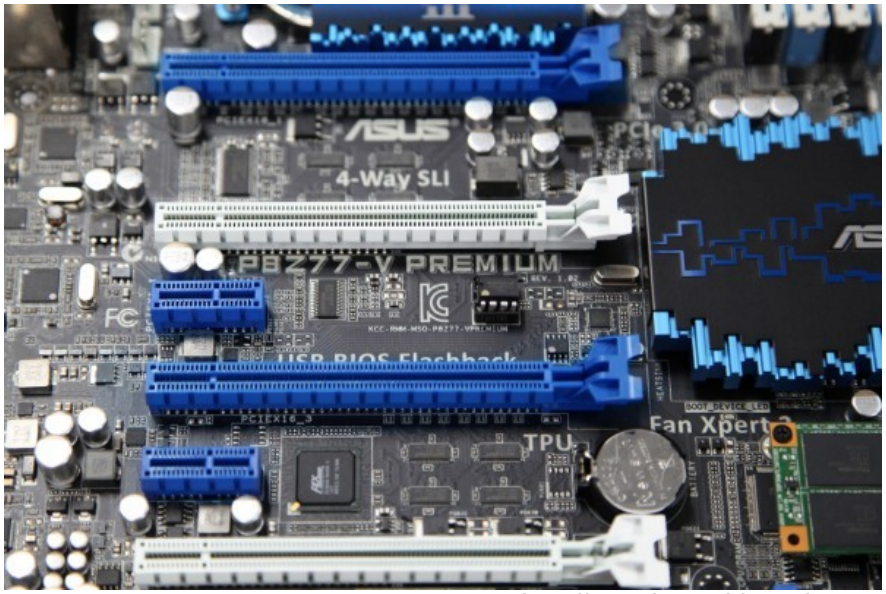
- **Bus locaux (local bus, front side bus)**
  - courts et haute vitesse
  - maximisent le débit mémoire
  - optimisés pour des transferts de blocs de cache
  - souvent propriétaires

  - exemples
    - Intel FSB : 64b, 266-400MHz, QDR
    - DDR(/2/3) interface

Types de bus

- **Backplane**
  - structure d'interconnexion dans un chassis
  - bus intermédiaire entre bus local et bus d'IO

  - exemples
    - Hypertransport : 32b, 200MHz à 2.6GHz, DDR, 2 canaux
    - PCI, PCI-express
    - ATA
    - ISA



http://www.journaldugeek.com

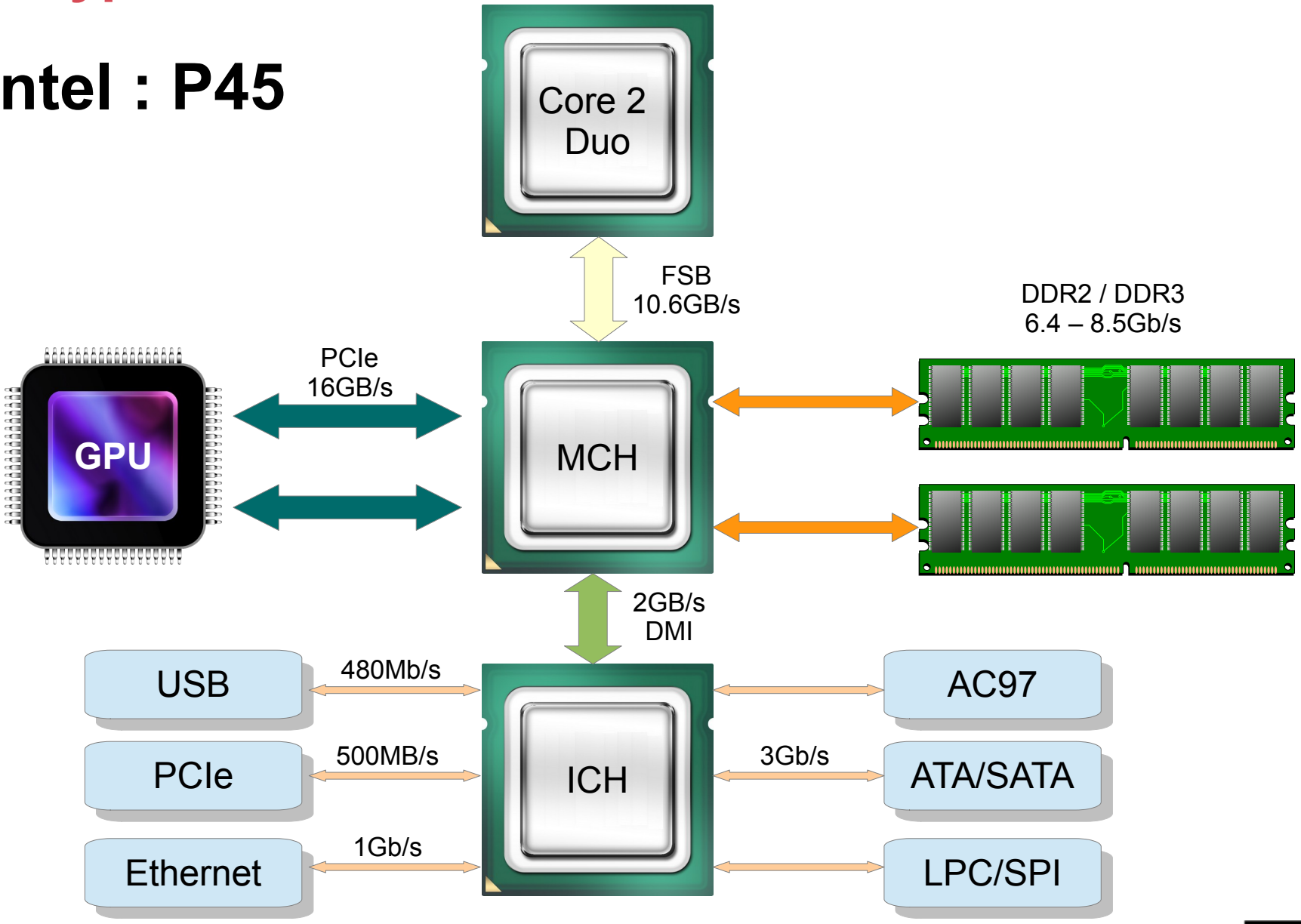
Types de bus

- **IO Bus**
  - plus longs, plus lents
  - beaucoup de types de bus car beaucoup de types d'IO
  - connectés au bus local et/ou à un backplane
  - exemples :
    - AGP, SATA, Thunderbolt
    - RS232, CAN, Flexray, I2C, SMBUS, SPI, USB
    - Ethernet, ADSL
    - 802.15.4, ZigBee, enOcean, Dash-7, WiFi, BT/BLE
    - DMX

http://www.journaldugeek.com

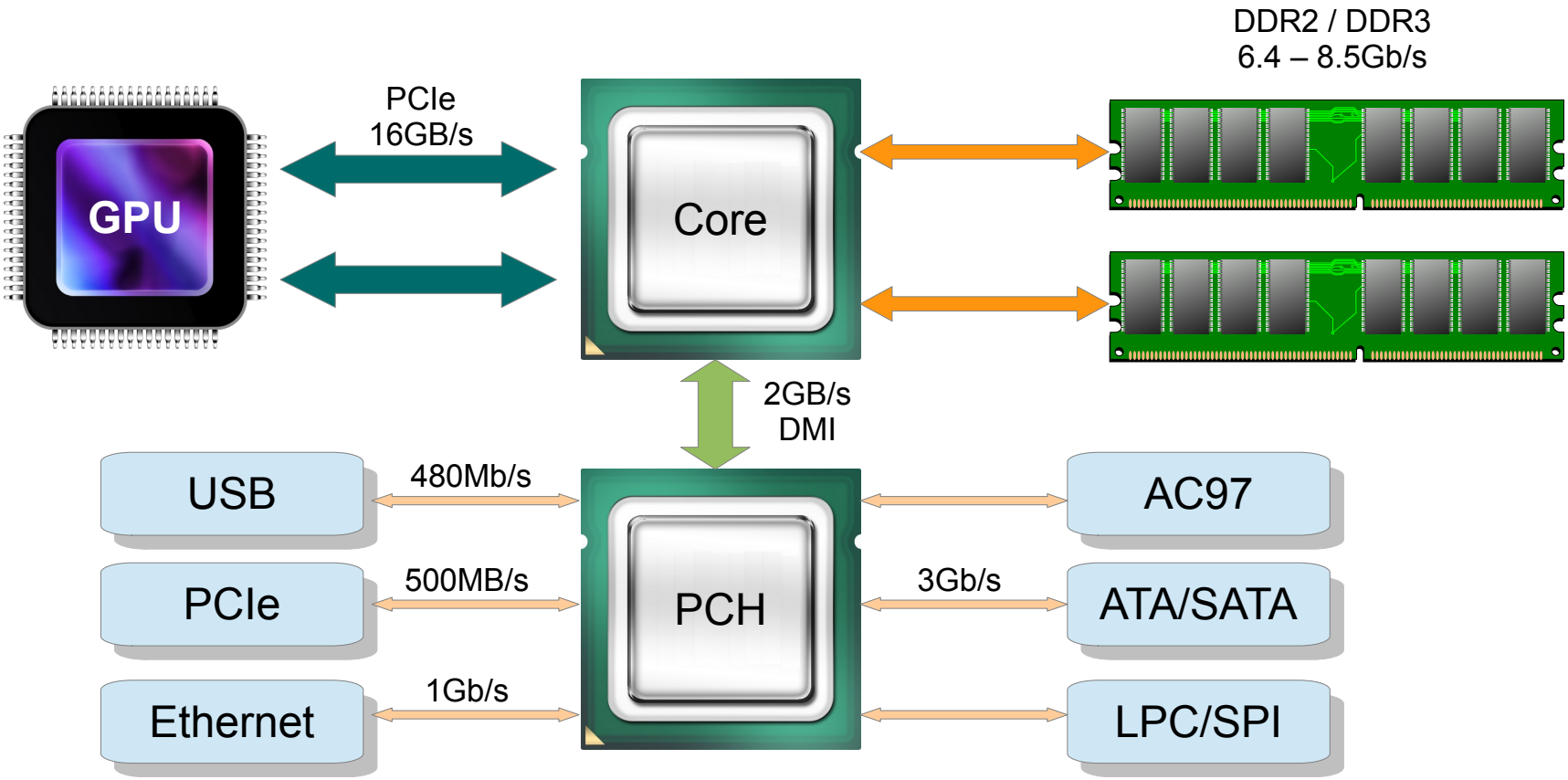
Types de bus

- **PC Intel : P45**



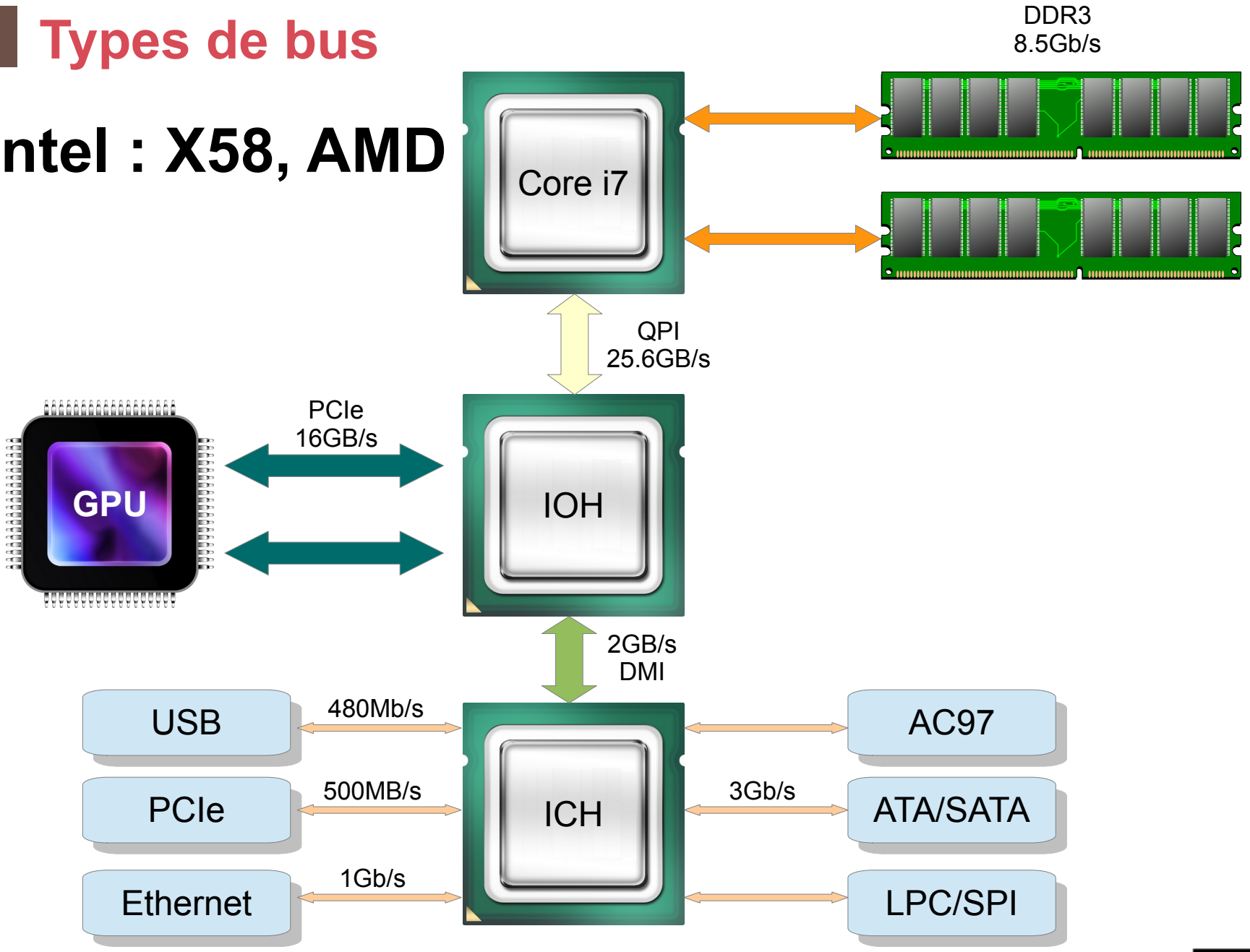
Types de bus

- **PC Intel : P55**



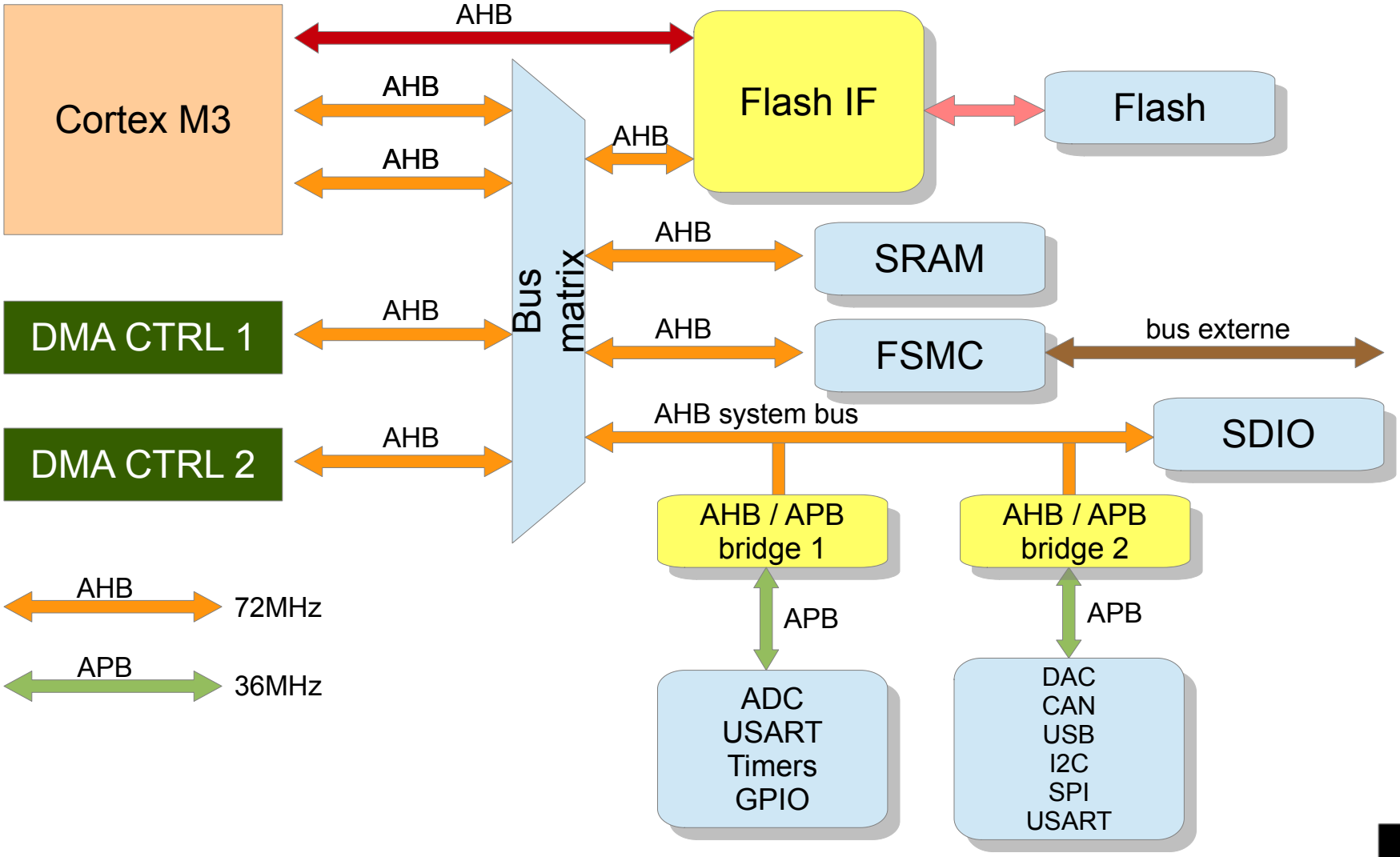
Types de bus

- PC Intel : X58, AMD

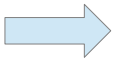


Types de bus

- **STM32F103 (Cortex M3)**



Plan



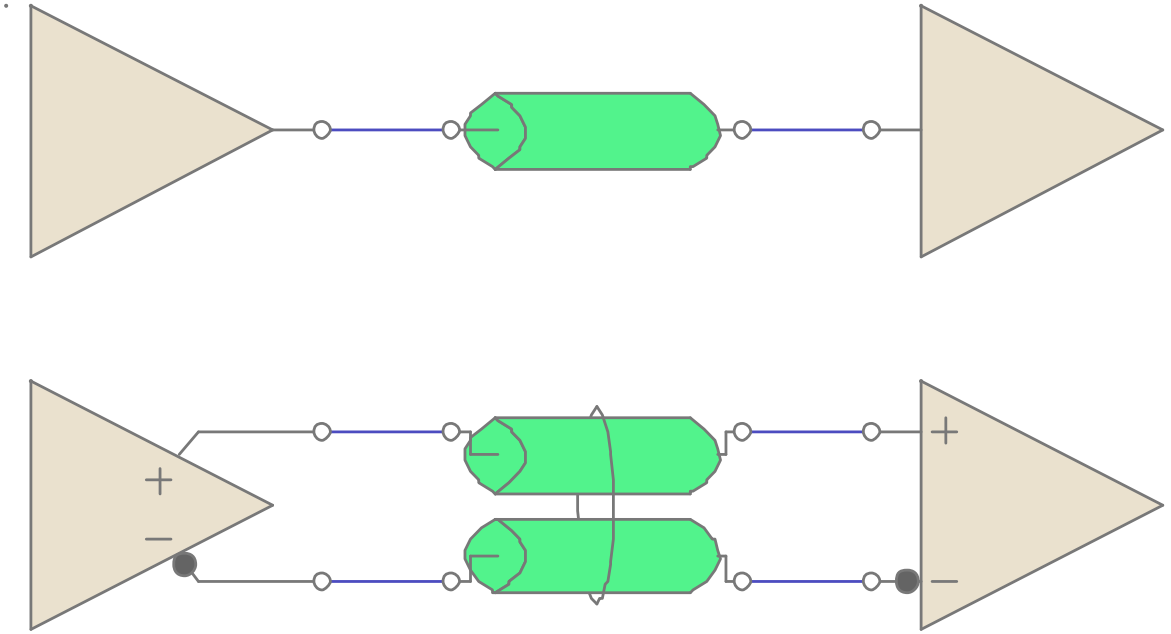
- Définition
- Structure physique
- Topologies
- Protocoles
- Arbitrage
- Timings
- Augmentation des performances
- Étude de bus embarqués usuels

Topologies

- **Topologies principales**
  - arrangement physique du bus dans l'espace
  - topologie : influe sur la qualité du signal
    - sera étudié en détail dans la partie sur l'intégrité du signal
  - topologies usuelles :
    - point à point
    - multipoint
      - multidrop
      - daisy-chain multidrop
      - tree
      - star

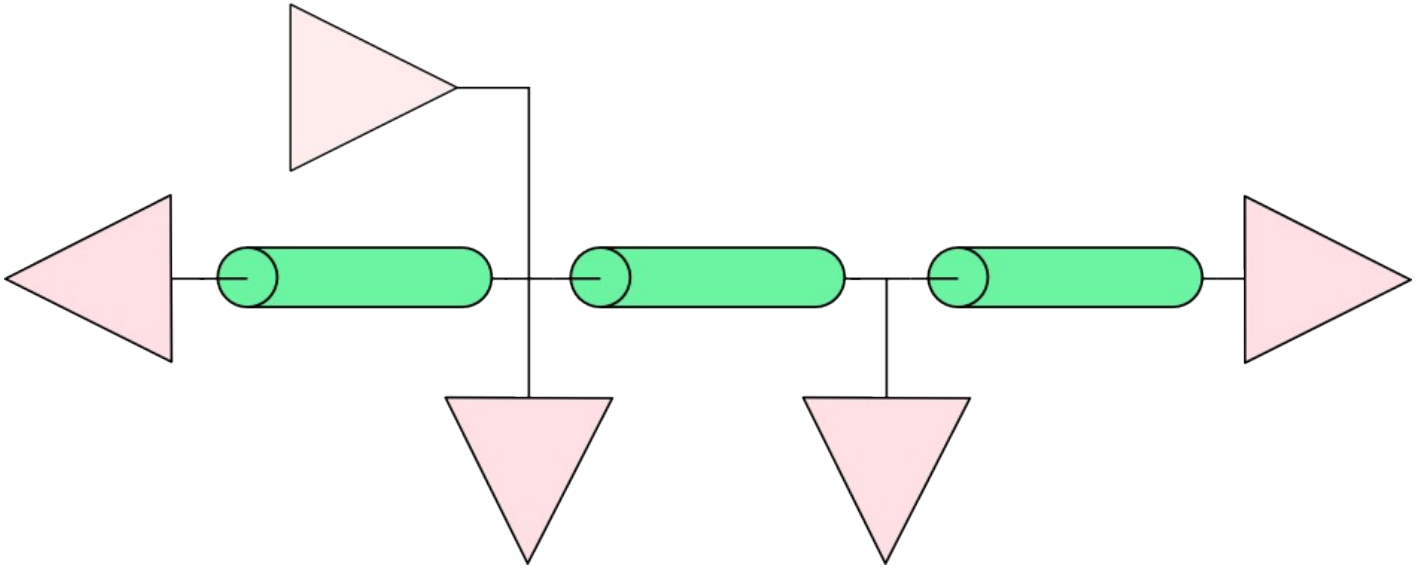
Topologies

- **Topologies principales**
  - point à point
    - single ended
    - différentiel



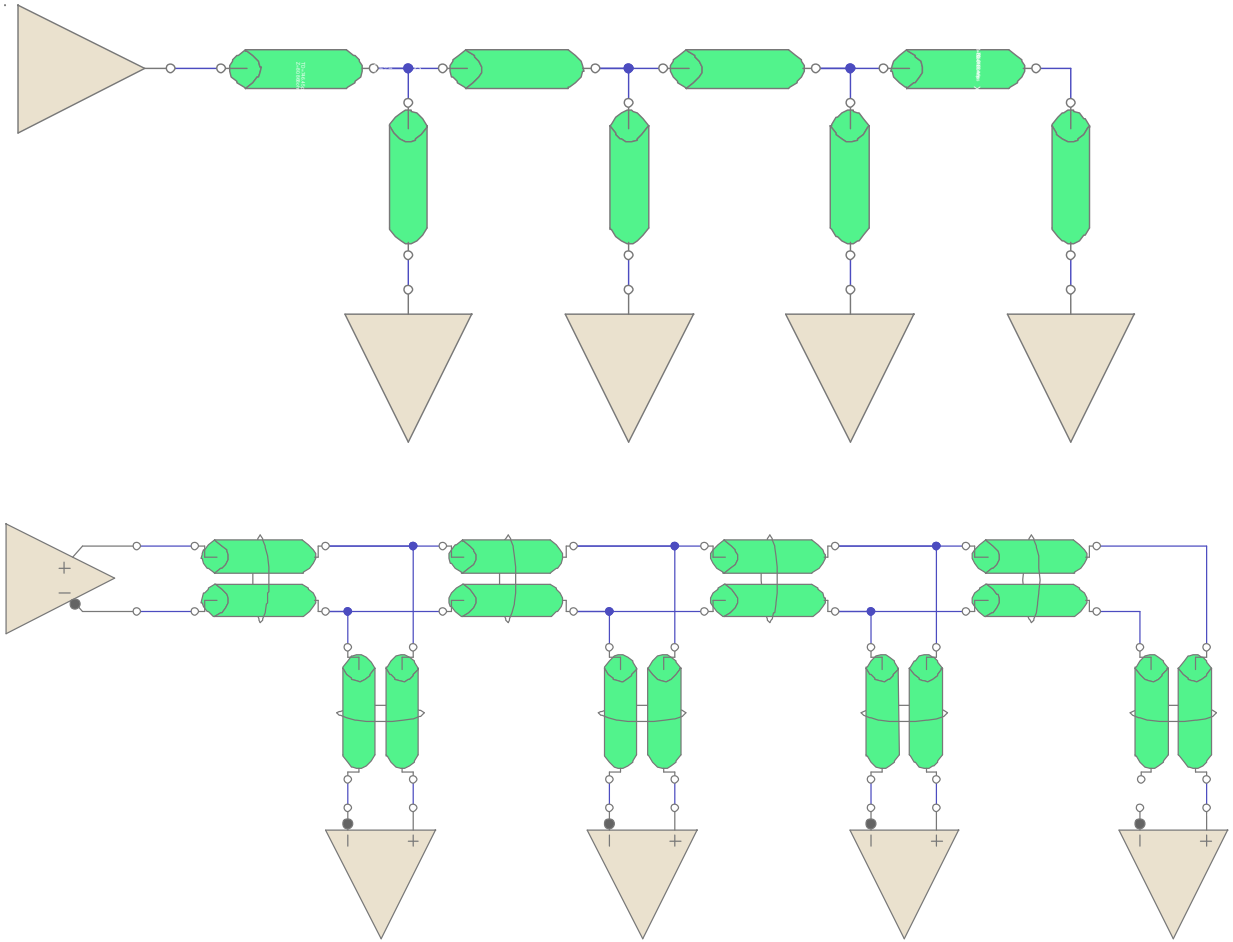
Topologies

- **Topologies principales**
  - multidrop



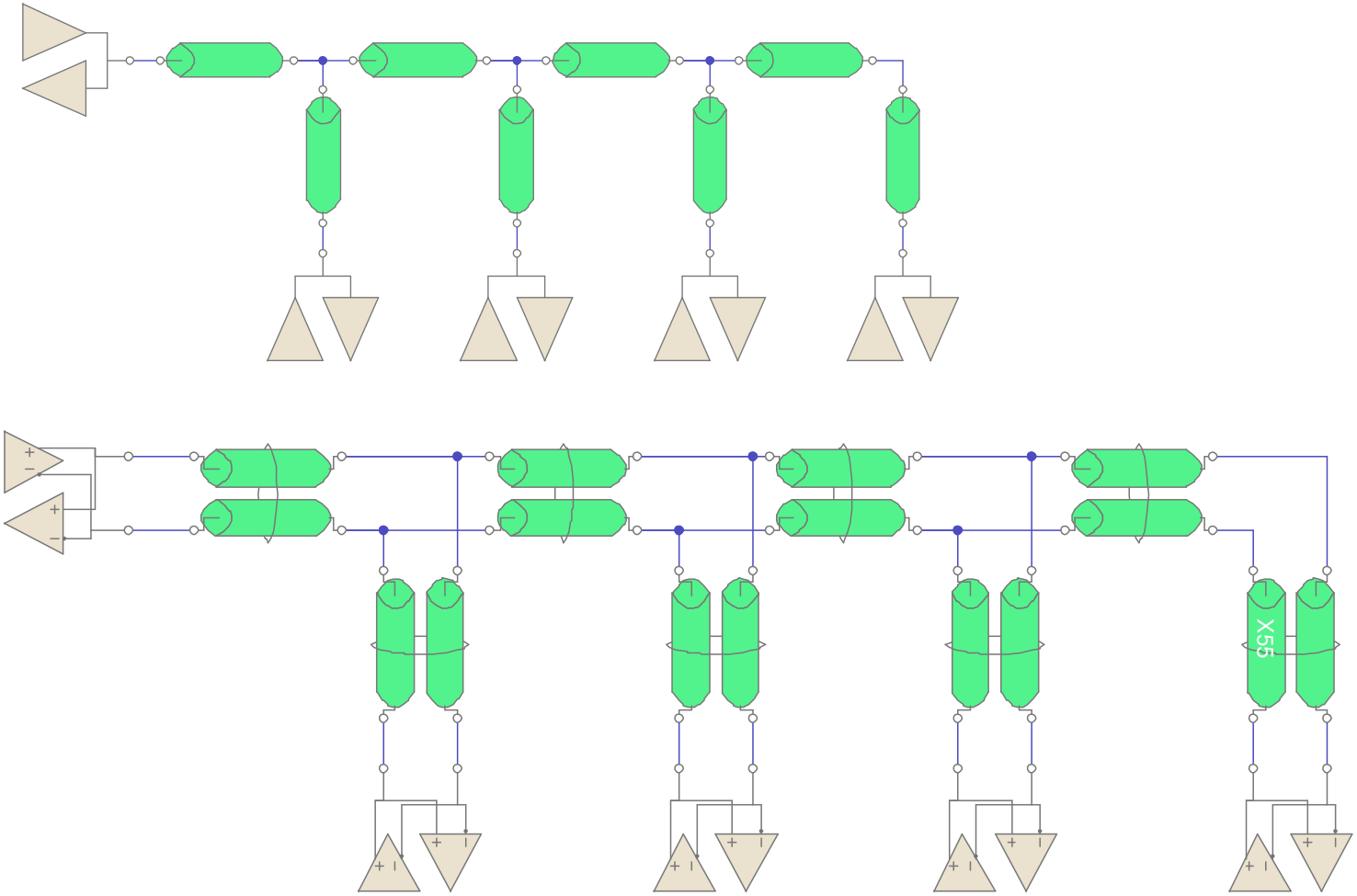
Topologies

- **Topologies principales**
  - multidrop daisy-chain



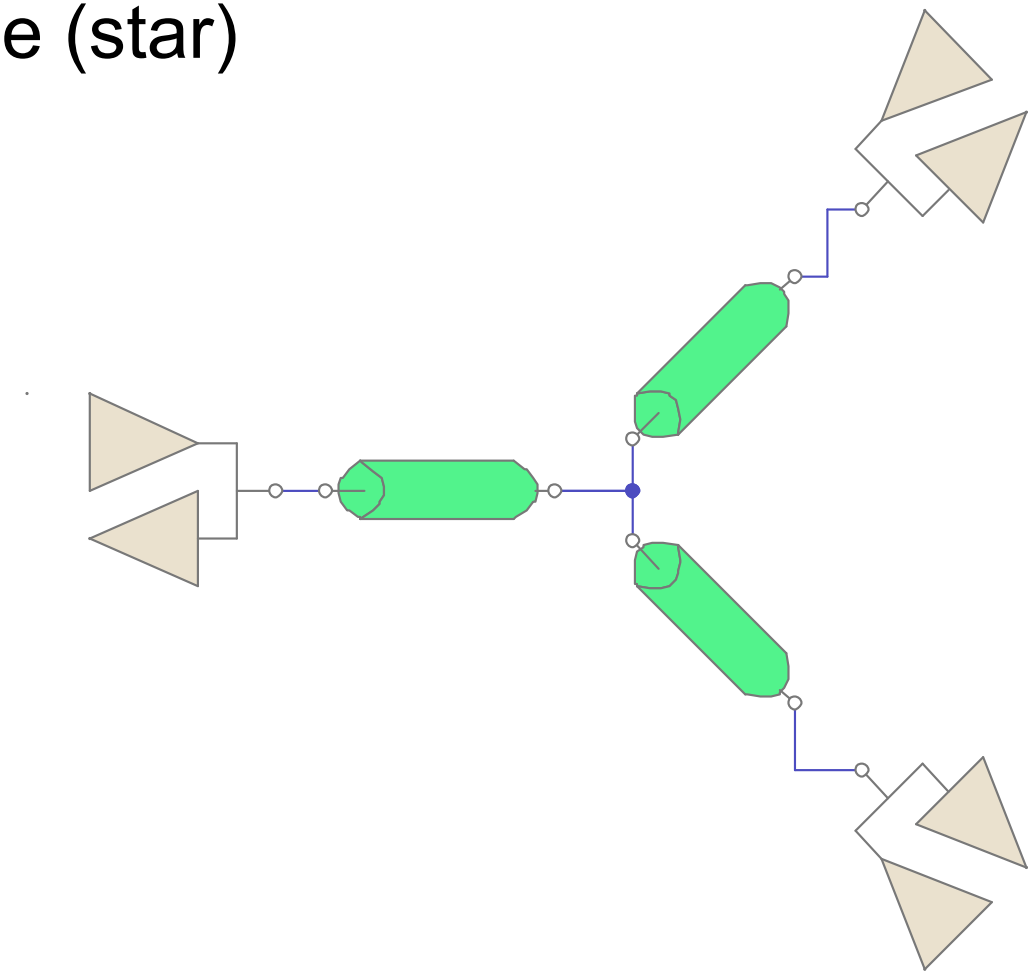
Topologies

- **Topologies principales**
  - multipoint daisy chain



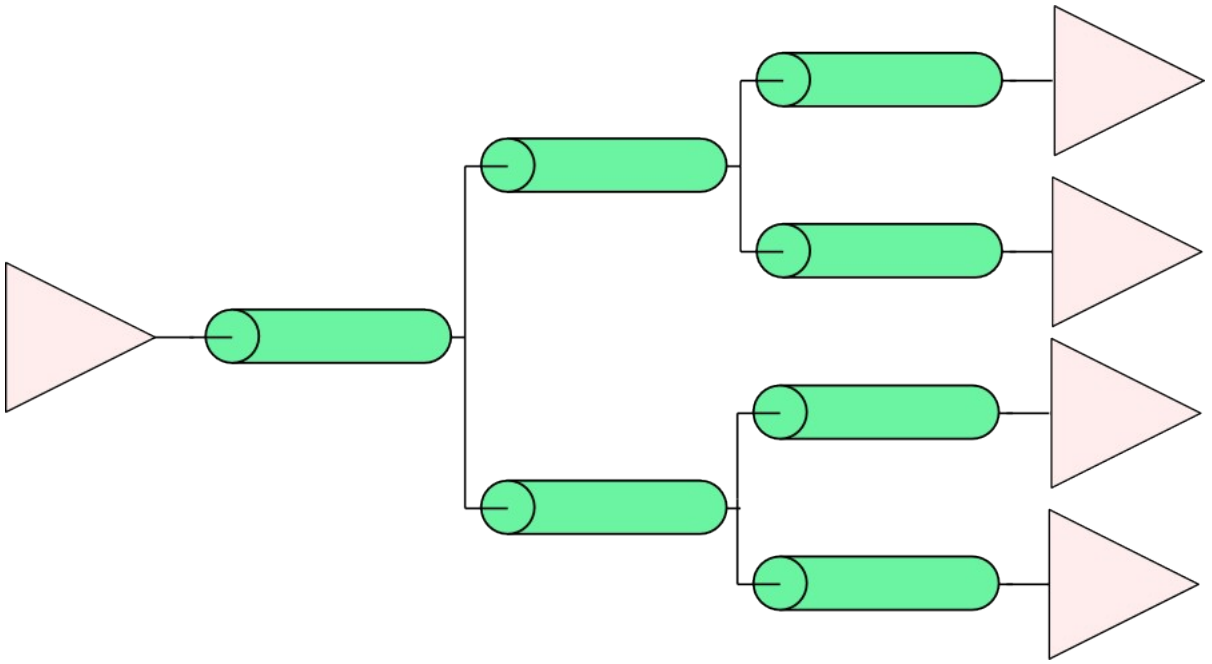
Topologies

- **Topologies principales**
  - étoile (star)

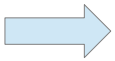


Topologies

- **Topologies principales**
  - arbre (tree)



Plan



- Définition
- Structure physique
- Topologies
- Protocoles
- Arbitrage
- Timings
- Augmentation des performances
- Étude de bus embarqués usuels

Protocole

- **Protocole**
  - Suite de transactions
    - transactions
      - un échange complet
      - formé de messages
    - messages (flit)
      - plus petite unité logique d'information transmise
      - dure un ou plusieurs cycles
      - peut être nul : cycle d'arbitrage
    - cycle
      - transmission d'un phit (physical digit)

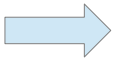
Protocole

- Exemple :
  - transaction : "lecture par un processeur d'une donnée 32 bits en mémoire à l'adresse 0x100", sur un bus synchrone 16 bits
    - message nul
      - cycles d'arbitrage entre les différents maîtres potentiels
    - message µP → mem : "je veux lire à telle adresse"
      - cycle de transmission de l'adresse (0x100) et du signal Read
    - message mem → µP : "la donnée est 0x44332211"
      - cycles d'attente
      - cycle de réponse : la première partie de la donnée vaut 0x2211
      - cycle de réponse : la deuxième partie de la donnée vaut 0x4433
      - cycle mort

Protocole

- **Rôles**
  - définitions :
    - *maître* : un nœud capable d'initier une transaction.
    - *esclave* : un nœud ne possédant pas cette possibilité.
    - à un instant t, plusieurs maîtres peuvent vouloir commencer une transaction en même temps. Sur un bus partagé, si on cherche à éviter les collisions, il est nécessaire de choisir celui qui aura le droit de le faire.
    - *arbitrage* : choix, parmi plusieurs maîtres souhaitant initier une transaction, de celui qui initiera la prochaine.
    - pour cette transaction,
      - le maître choisi est appelé *initiateur*
      - les autres nœud sont appelés *cibles* (qu'ils soient maîtres ou esclaves)

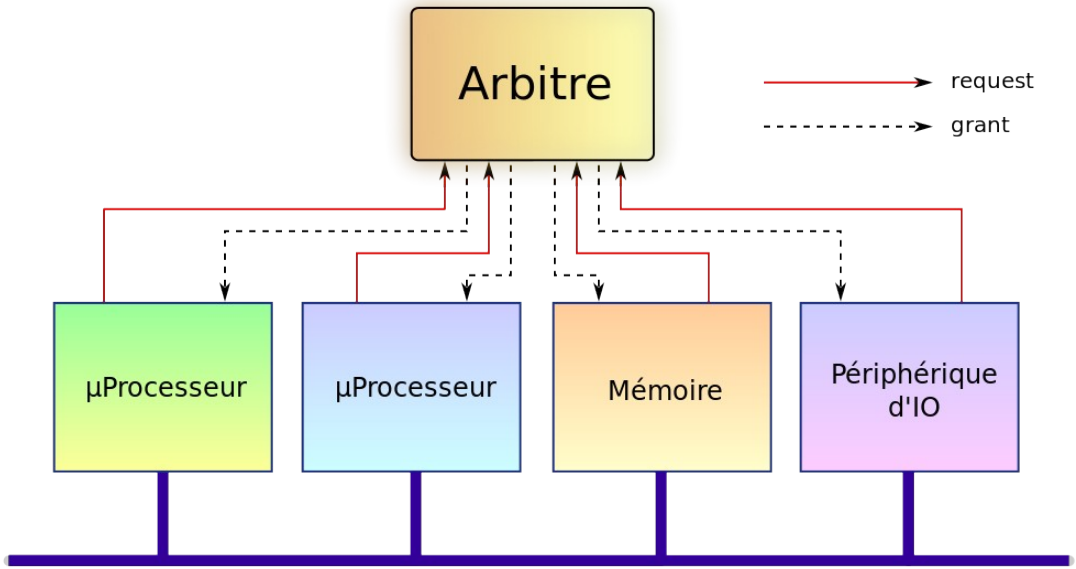
Plan



- Définition
- Structure physique
- Topologies
- Protocoles
- Arbitrage
  - définitions, propriétés
  - centralisé (à priorités)
  - décentralisé
- Timings
- Augmentation des performances
- Bus embarqués usuels

Arbitrage

- **Arbitres**
  - doivent décider de qui sera le prochain initiateur, sous contraintes de :
    - priorités : servir en premier ce qui en ont le plus besoin
    - équité : éviter les famines / dead-lock
  - Exemple : arbitre centralisé

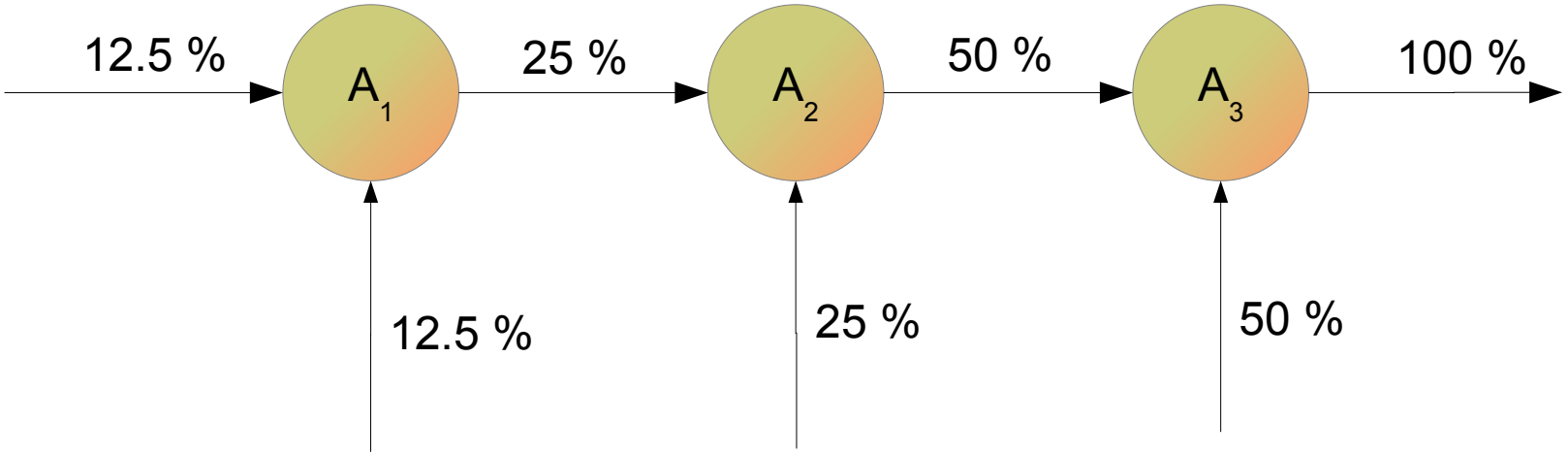


Arbitrage

- **Équité**
  - propriété clef des arbitres
  - classification :
    - *équité faible* : chaque requête sera honorée en un temps fini
    - *équité forte* : les requêtes seront honorées de façon équitable
      - cas particulier : premier arrivé, premier servi (FIFO)
    - *équité forte pondérée* : chaque requête sera honorée avec une fréquence proportionnelle à sa priorité

Arbitrage

- **Équité**
  - l'équité forte est une propriété locale
  - un assemblage d'arbitre fortement équitables ne produit pas forcément un système fortement équitable

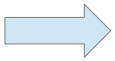


Arbitrage

- **L'arbitrage peut être :**
  - centralisé (radial)
    - un arbitre se charge de récupérer les demandes et d'attribuer les autorisations
    - souplesse des mécanismes de priorité
    - passage à l'échelle difficile
  - distribué
    - chaque nœud dispose de son propre arbitre
    - à ne pas confondre avec un arbitre répliqué (VAX SBI Bus)
    - passage à l'échelle plus simple
  - mixte (daisy-chain)
    - un arbitre central fait une première sélection
    - le reste est fait de façon distribuée

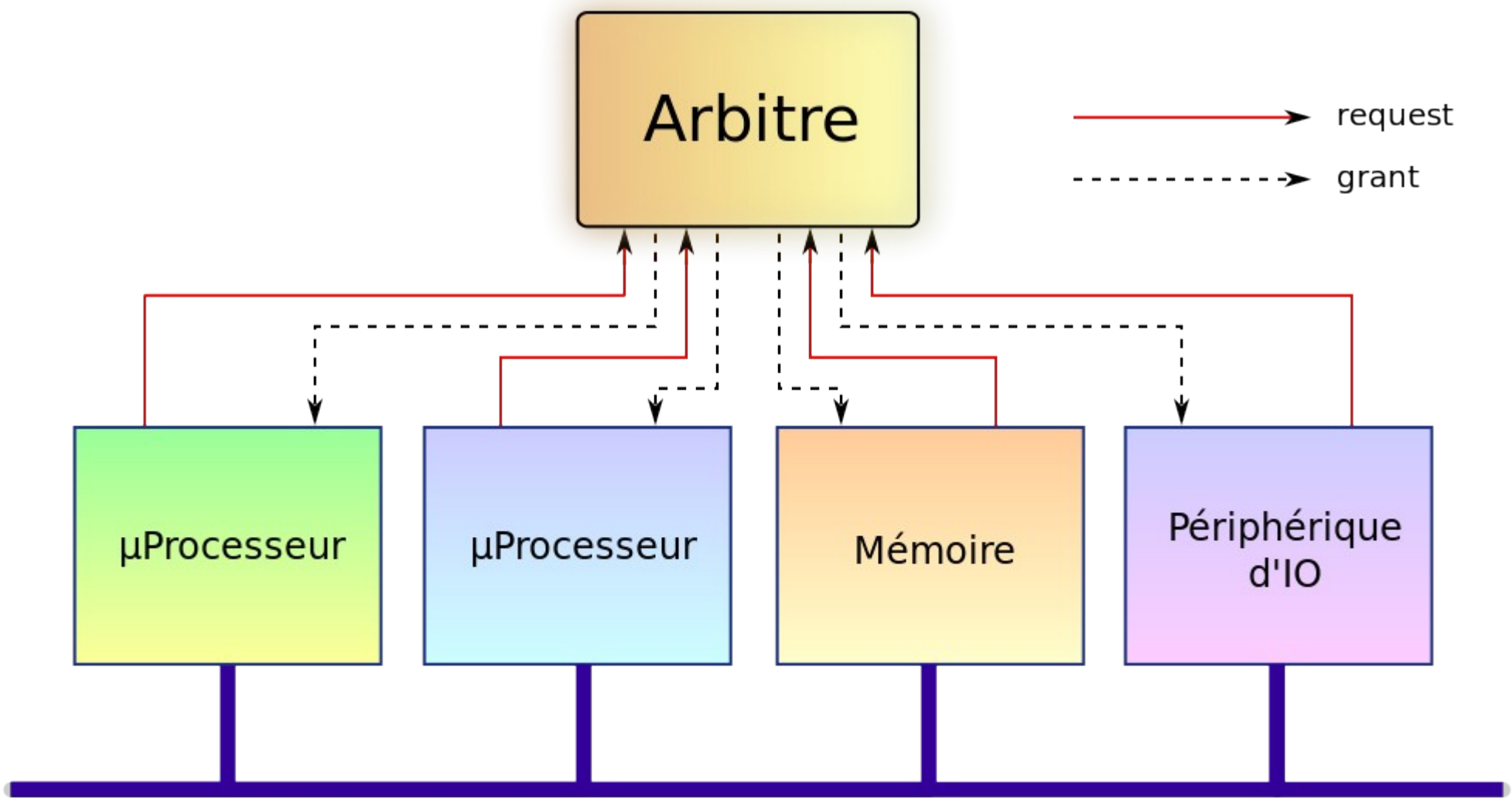
Plan

- Définition
- Structure physique
- Topologies
- Protocoles
- Arbitrage
  - définitions, propriétés
  - centralisé (à priorités)
  - décentralisé
- Timings
- Augmentation des performances
- Bus embarqués usuels

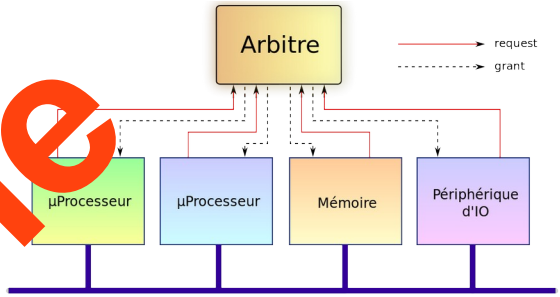


Arbitrage centralisé

- **Arbitre centralisé**



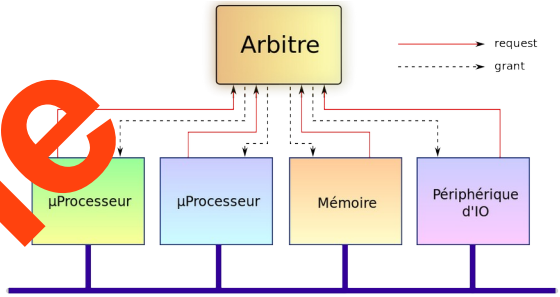
Arbitrage centralisé



- **Gestion des priorités**
  - priorités statiques
    - non équitable...
  - priorités variables
    - priorités cycliques
    - round-robin
    - LRU
    - FIFO
  - pondération des priorités

Hors programme

Arbitrage centralisé

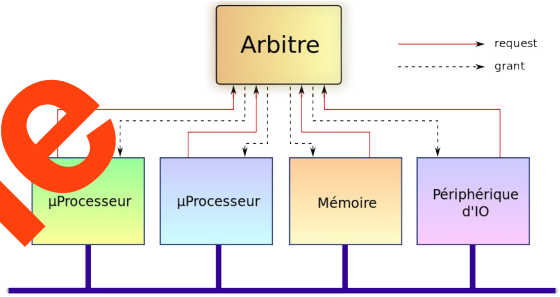


- **Gestion des priorités**
  - priorités statiques
    - i < j → i plus prioritaire que j
    - r_i : lignes de requêtes
    - g_i : lignes de grant

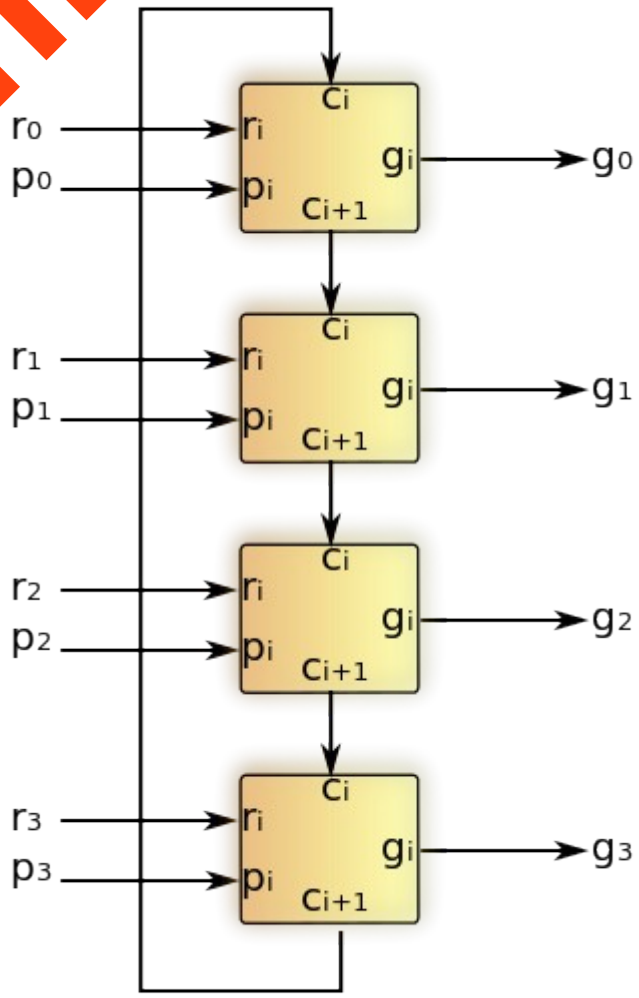
  - architecture ?
  - inconvénients ?

Hors programme

Arbitrage centralisé

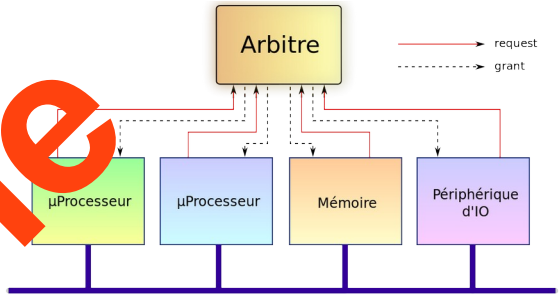


- **Gestion des priorités**
  - priorités itératives
    - p_i : one hot, indiquant la ligne de plus haute priorité
    - les priorités décroissent ensuite de manière cyclique
    - architecture ?
    - comment éviter la boucle de logique combinatoire ?
    - reste à générer les p_i de façon intelligente : c'est l'objectif des prochains slides !

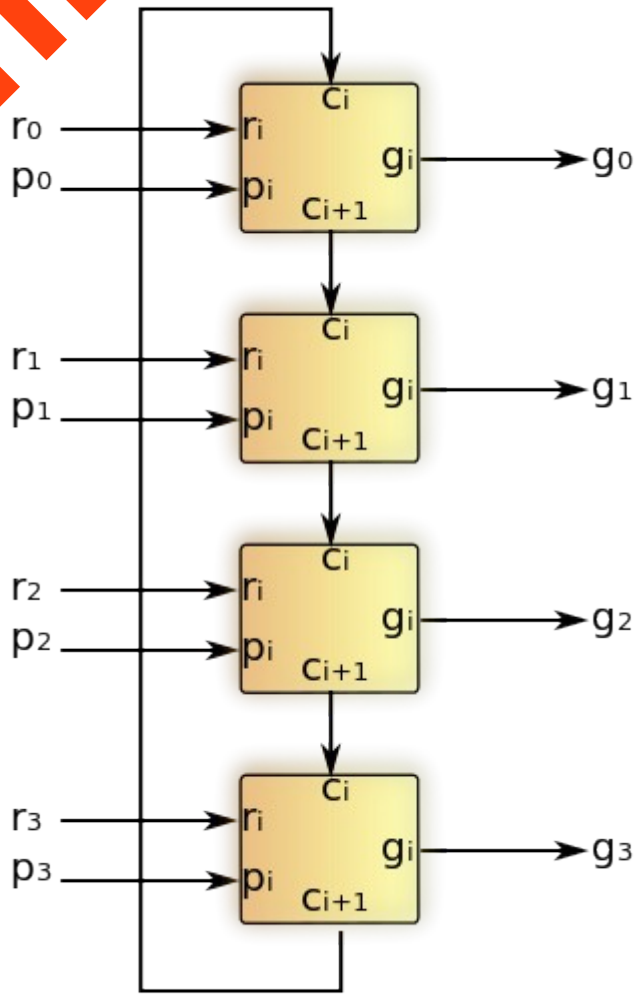


Hors programme

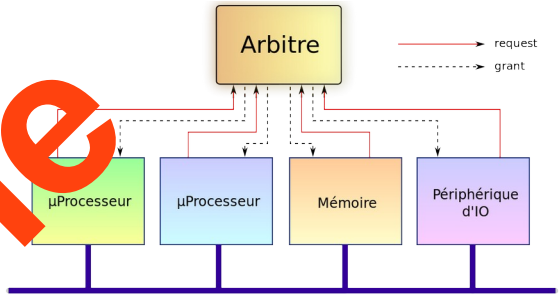
Arbitrage centralisé



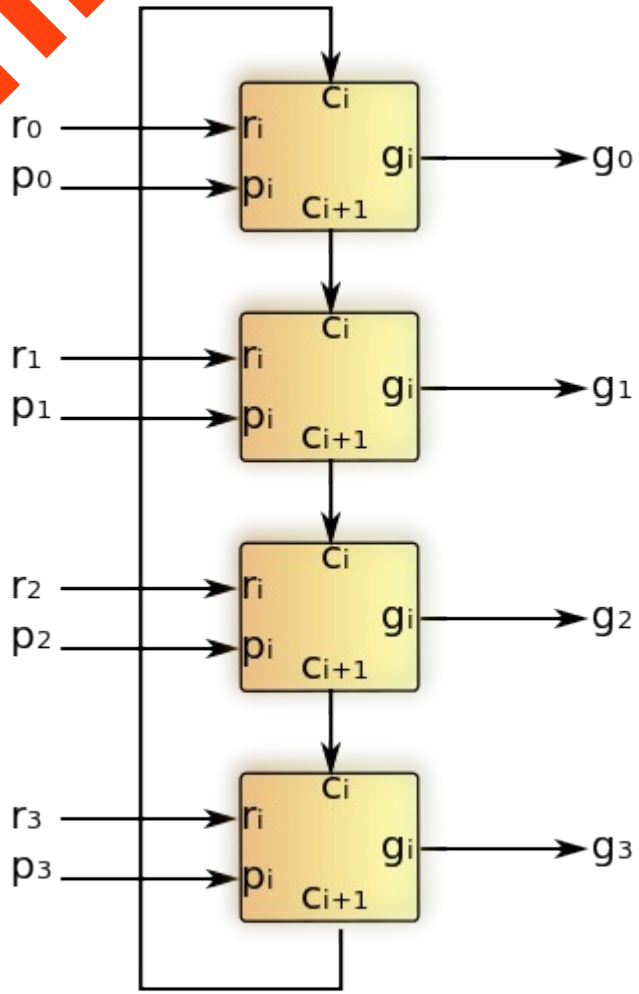
- **Gestion des priorités**
  - priorités itératives cycliques
    - p_i : cycliques et 1-hot
    - architecture du générateur de p_i ?
    - inconvénients du système ?



Arbitrage centralisé

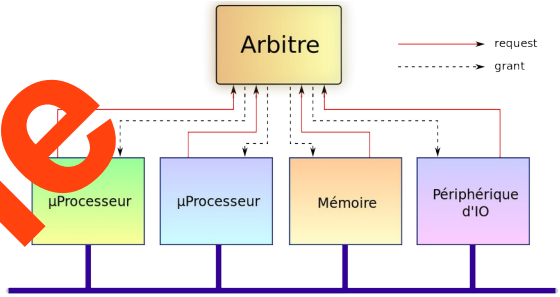


- **Gestion des priorités**
  - priorités itératives round-robin
    - on génère les p_i de façon à ce que la dernière requête servie devienne la moins prioritaire
    - architecture du générateur de p_i ?
    - équité ?



Hors programme

Arbitrage centralisé

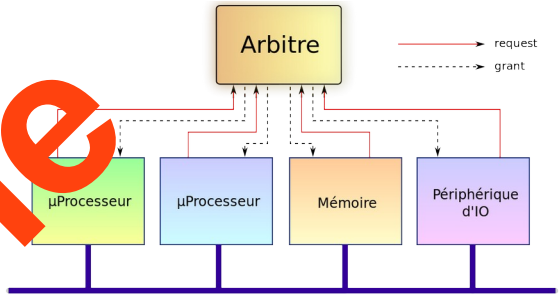


- **Gestion des priorités**
  - LRS
    - la requête la plus prioritaire est celle qui a été servie il y a le plus longtemps
    - en quoi cela diffère-t-il du round-robin itératif ?
    - temps moyen d'attente pour une tâche rare ?
    - équité ?

Requêtes	RR itératif	LRS
0, ?		
0, 2		
0, 1, 2		
0, 1, 2, 3		
0, 1, 2, 3		
0, 1, 2, 3		
0, 1, 2, 3		

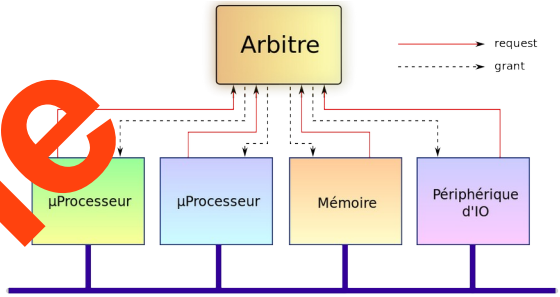
Hors programme

Arbitrage centralisé

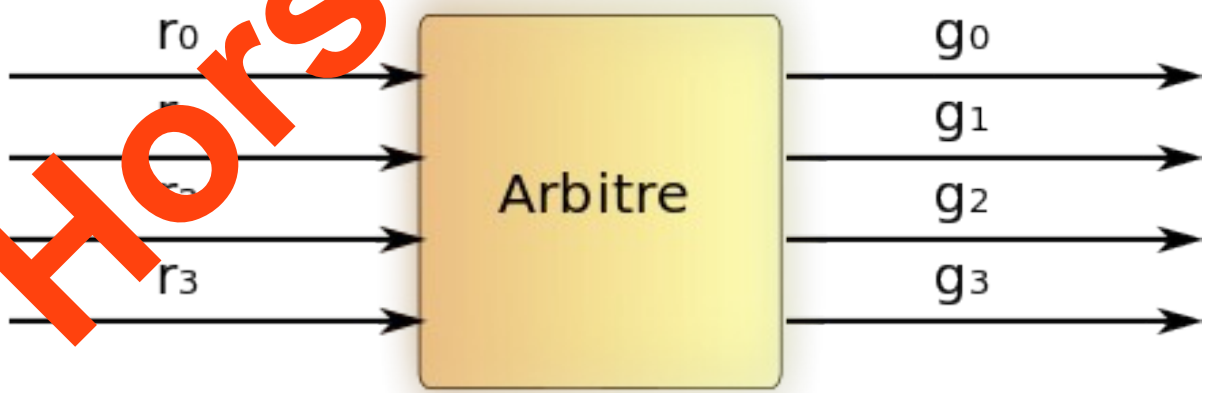


- **Gestion des priorités**
  - LRS : architecture 1
    - la priorité de chaque requête est stockée dans un registre
    - on maintient à jour ces priorités
  - Architecture du système ?

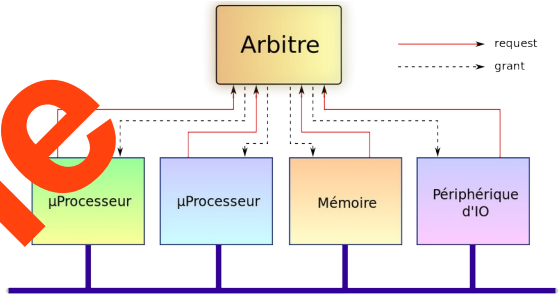
Arbitrage centralisé



- **Gestion des priorités**
  - LRS : architecture 2
    - on maintient une matrice de booléens W_{ij} indiquant si la requête i est plus prioritaire de la requête j
    - système de mise à jour des W_{ij}
    - montrer que si la colonne j possède un 1 sur une ligne d'une requête active i, alors g_j sera forcément nul
    - en déduire la génération des g_i

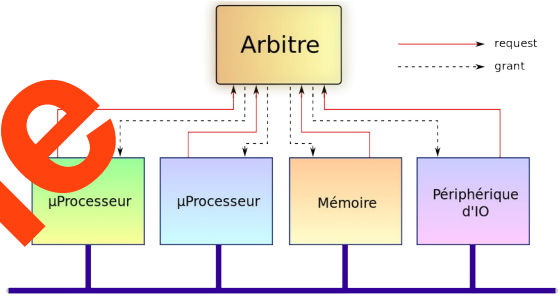


Arbitrage centralisé



- **Gestion des priorités**
  - LRS : architecture 2
    - que se passe-t-il si la matrice est initialisée avec des coefficients absurdes ?
    - proposer un moyen de détecter et corriger ces coefficients

Arbitrage centralisé

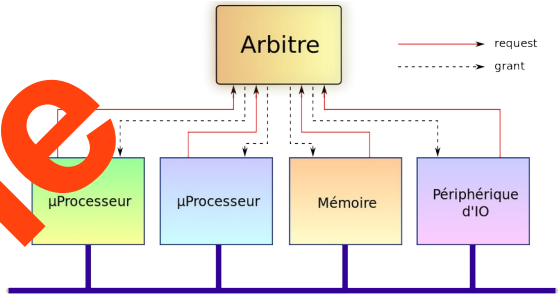


- **Gestion des priorités**
  - FIFO
    - premier arrivé, premier servi
    - en quoi cela diffère-t-il de RR itératif et LRU ?
    - quel type d'équité cet arbitrage garantit-il ?

Requêtes	RR itératif	LRU	FIFO
0, 2			
0, 2			
0, 1, 2			
0, 1, 2, 3			
0, 1, 2, 3			
0, 1, 2, 3			
0, 1, 2, 3			

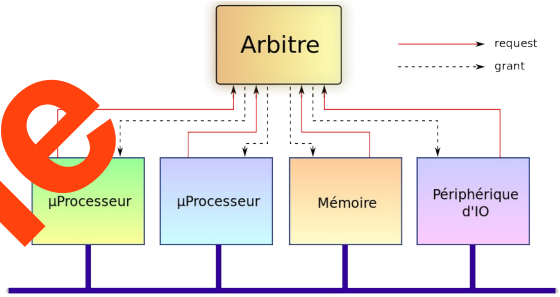
Requêtes	RR itératif	LRU	FIFO
0, 1, 3			
0, 1, 3			
0, 1, 3			
0, 1, 2, 3			
0, 1, 2, 3			
0, 1, 2, 3			
0, 1, 2, 3			

Arbitrage centralisé



- **Gestion des priorités**
  - FIFO
    - en reprenant l'architecture de LRS 1, proposer une architecture pour cet arbitre FIFO

Arbitrage centralisé

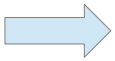


- **Gestion des priorités**
  - pondération des priorités
    - on attribue à chaque nœud un poids : w_i
    - le temps est séparé en paquets de $\sum w_i$ cycles
    - chaque nœud se voit attribuer w_i cycles dans chaque paquet
    - comment rajouter cette propriété aux arbitres précédents ?
    - inconvénients ?

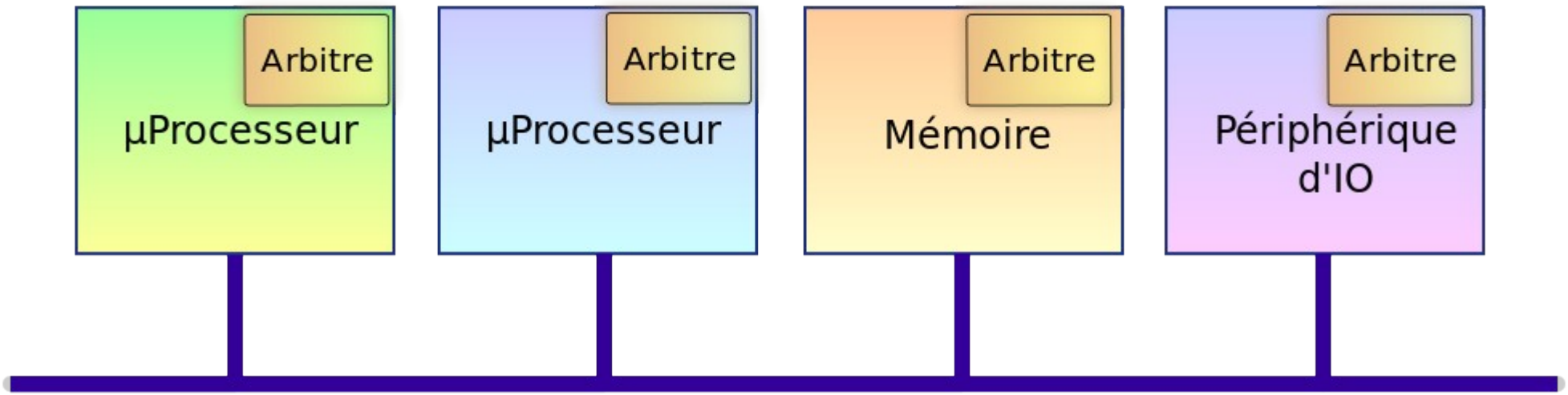
Hors programme

Plan

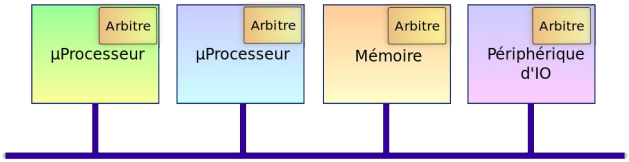
- Définition
- Structure physique
- Topologies
- Protocoles
- Arbitrage
  - définitions, propriétés
  - centralisé (à priorités)
  - décentralisé
- Timings
- Augmentation des performances
- Bus embarqués usuels



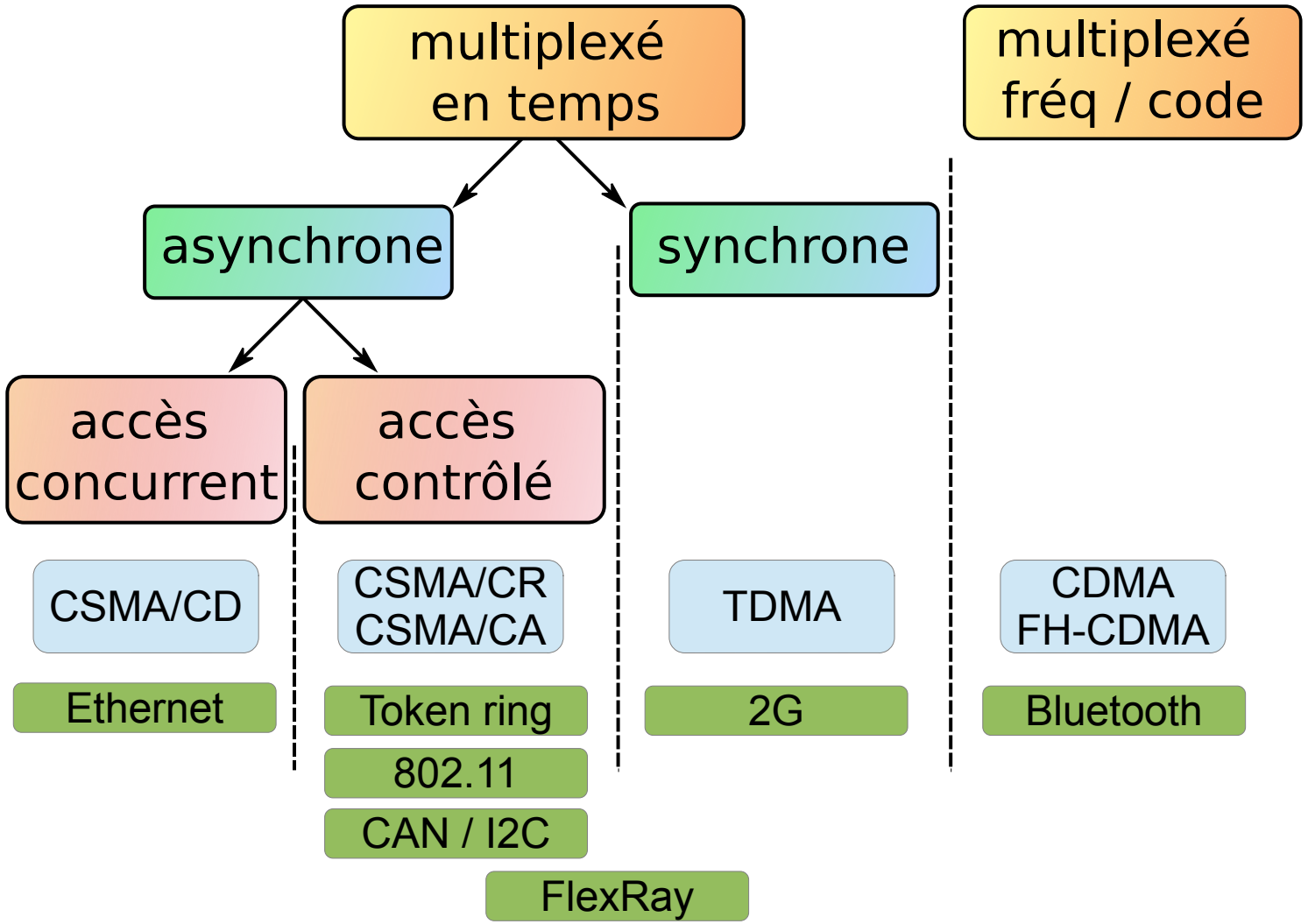
- **Arbitrage distribué**



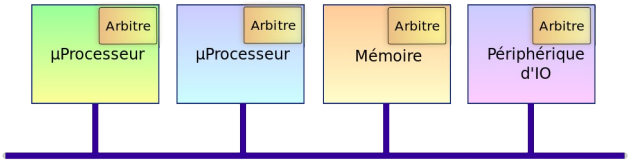
Arbitrage distribué



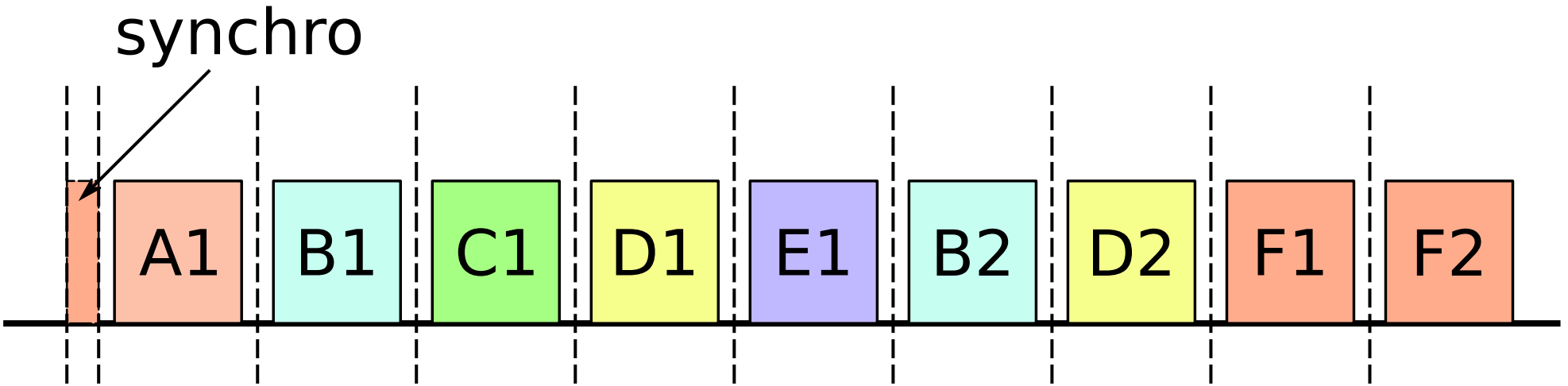
- **Arbitrage distribué**



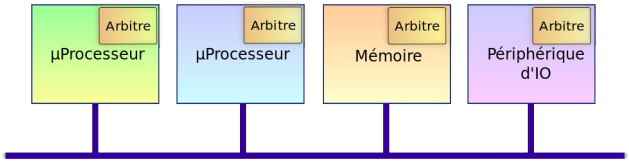
Arbitrage distribué



- TDMA
  - le temps est découpé en paquets (time slots)
  - ces paquets sont alloués statiquement à chaque nœud pour ses communications

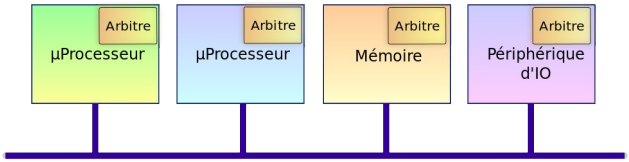


Arbitrage distribué

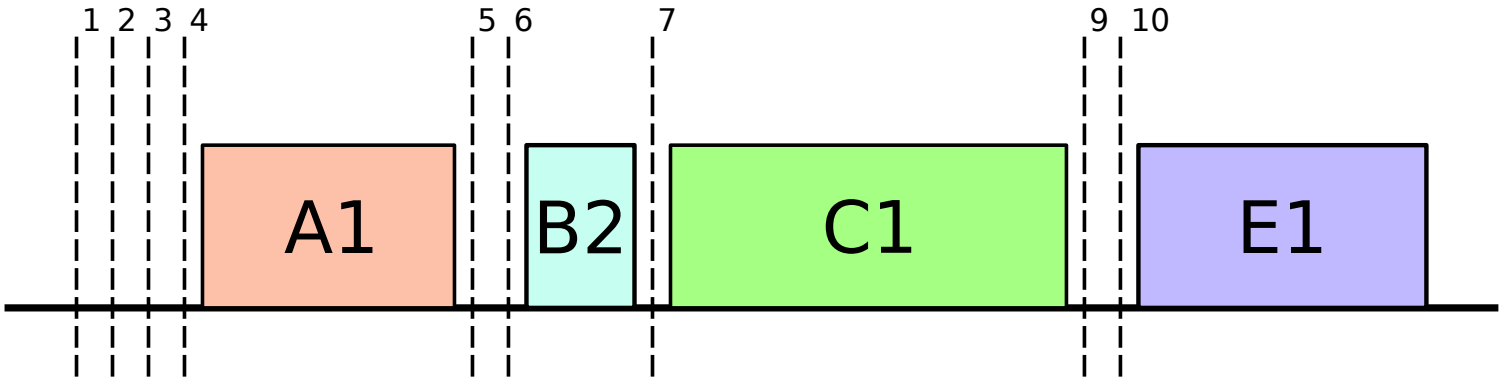


- TDMA
  - il est nécessaire de synchroniser temporellement les nœuds entre eux
    - envoi d'un pattern particulier (appelée trame de synchronisation)
      - par un maître déclaré statiquement
      - par un maître déclaré dynamiquement
    - envoi d'un pattern particulier à chaque transmission
  - exemples :
    - TTP, 2G
    - partie statique de FlexRay, TTCAN
  - inconvénient :
    - les slots sont alloués statiquement
      - pire cas
      - difficulté de modification du nombre de nœuds
    - en fonctionnement normal : bus sous-utilisé

Arbitrage distribué

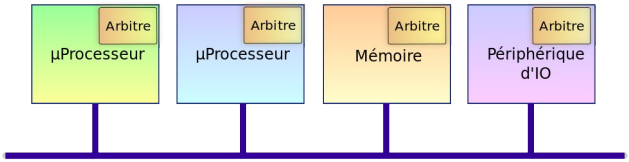


- CSMA/CA : Flexible TDMA (FTDMA)
  - Carrier Sense Multiple Access / Collision Avoidance
  - minislots
    - alloués statiquement à chaque nœud
    - ne déterminent que les débuts de trame
    - le comptage des minislots est interrompu pendant les communication
    - il reprend une fois la communication terminée



  - exemple :
    - 802.11 (Wireless Ethernet)
    - partie dynamique de FlexRay

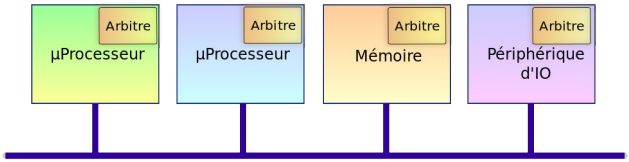
Arbitrage distribué



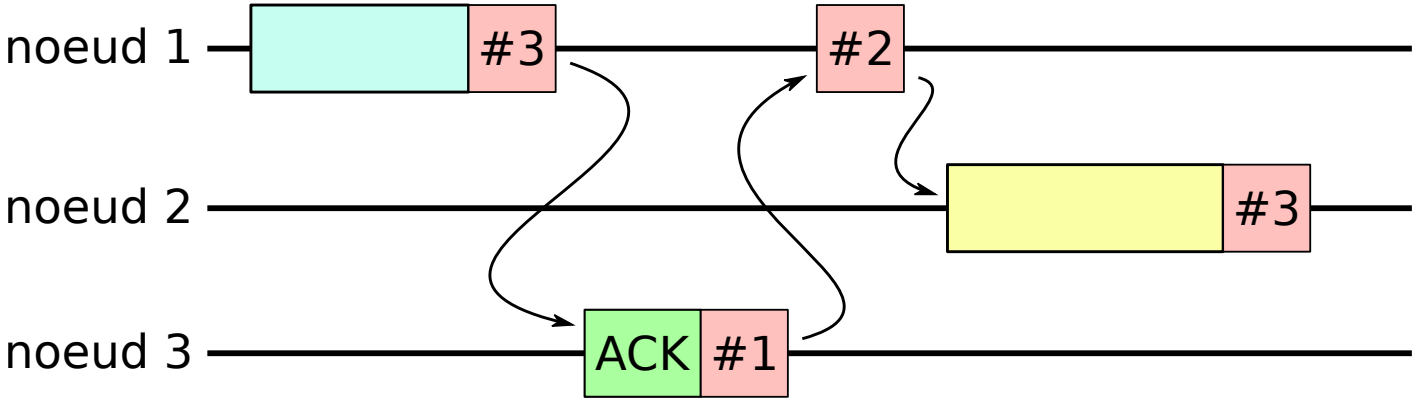
- CSMA/CA : protocoles à jetons (token protocols)
  - un jeton virtuel unique existe dans le réseau
  - seul celui qui dispose du jeton a le droit de transmettre
  - une fois la communication terminée, le possesseur du jeton le passe au nœud suivant

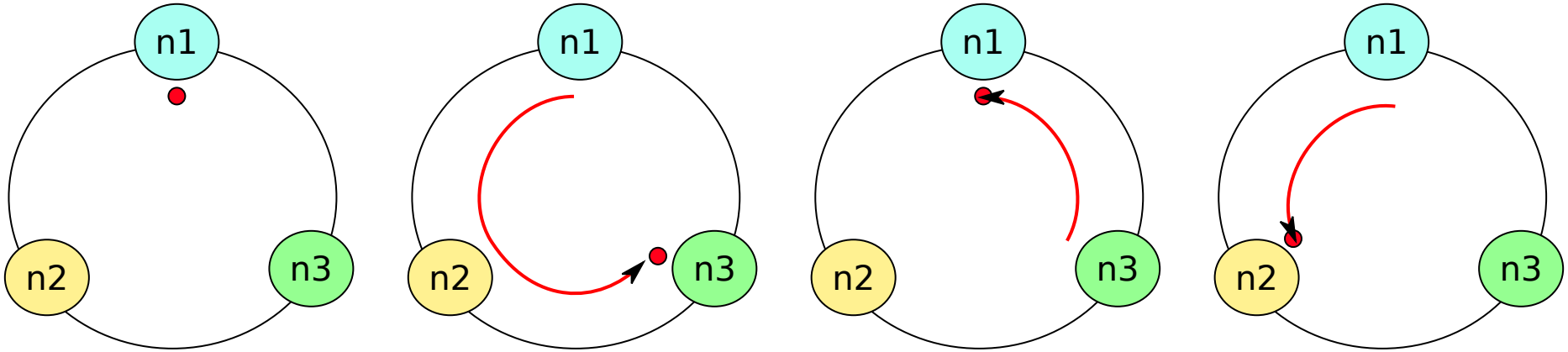
  - remarques :
    - le passage du jeton peut s'effectuer :
      - à l'aide d'un message spécial sur le bus
      - à l'aide de lignes dédiées
    - le passage du jeton peut impliquer des messages vides
    - le jeton peut sauter des participants
    - prévoir un mécanisme de récupération en cas de perte de jeton

Arbitrage distribué



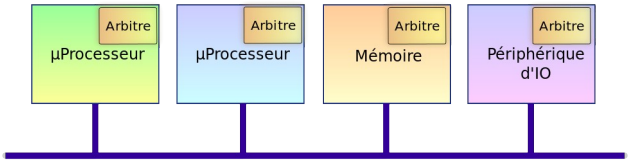
- CSMA/CA : protocoles à jetons (token protocols)



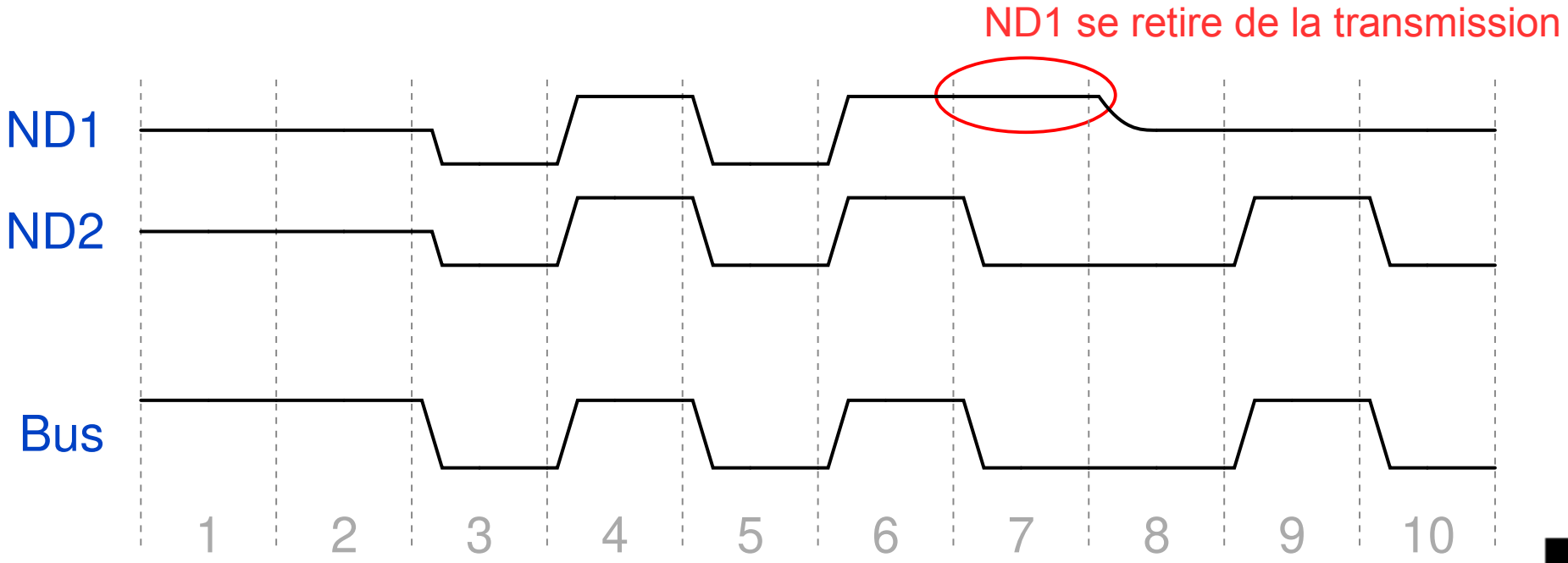


n1 a le jeton | n1 à n3 : blabla | n3 à n1 : ACK | n1 à n2 : à ton tour

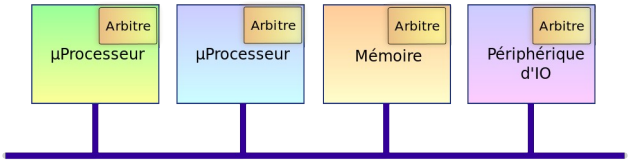
Arbitrage distribué



- CSMA/CR
  - Carrier Sense Multiple Access / Collision Resolution
  - principe : détecter et résoudre les collisions au vol
    - chaque nœud commence à transmettre comme s'il était seul
    - et regarde ce qui est effectivement transmis
    - en cas de non-concordance, l'une des parties se retire de la transmission

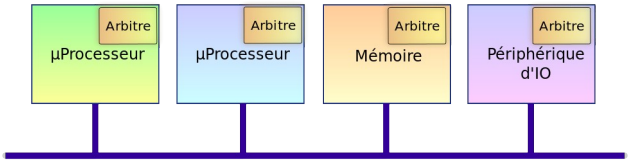


Arbitrage distribué

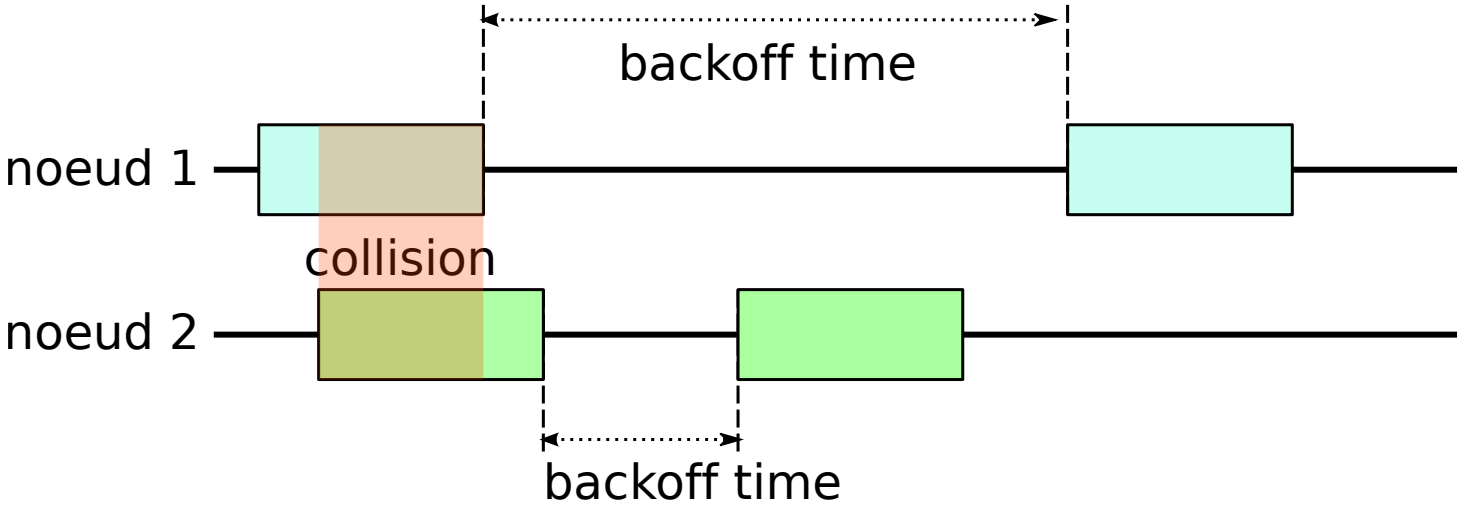


- CSMA/CR
  - remarques
    - le bus doit accepter les conflits (open drain, …)
    - un état doit primer sur l'autre : états dominants / récessifs
    - on peut prioriser facilement les messages (comment ?)
  - exemples :
    - CAN
    - I2C

Arbitrage distribué

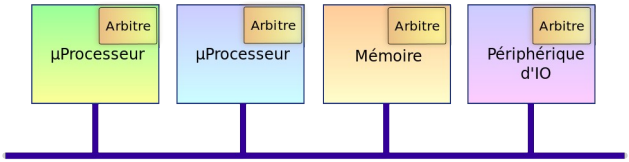


- CSMA/CD
  - Carrier Sense Multiple Access / Collision Detection
  - principe : détecter les collisions, s'arrêter et ré-essayer plus tard
    - en cas de collision, arrêt des transmission
    - re-essai après un temps aléatoire (backoff period)

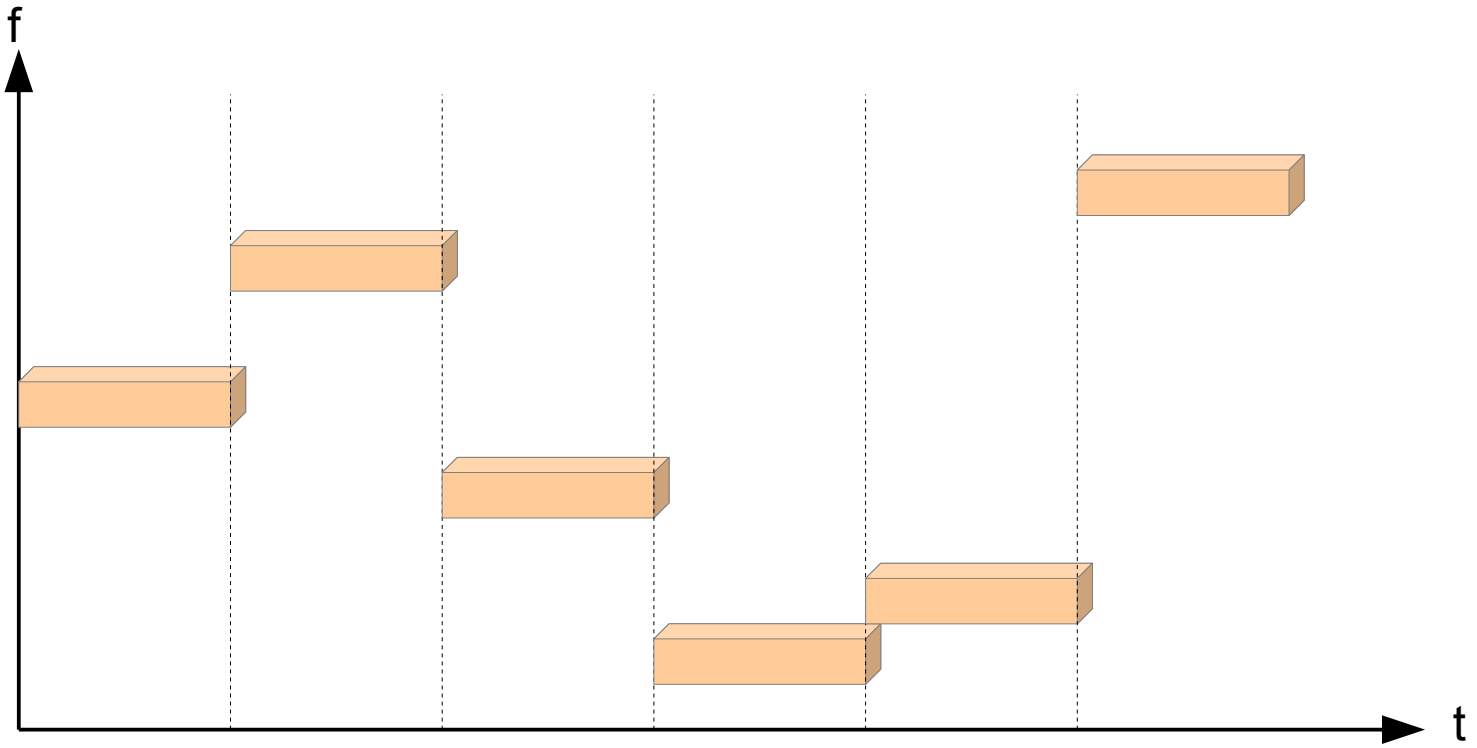


  - si un nœud sait détecter qu'un autre nœud est en train de transmettre, comment une collision peut-elle arriver ?

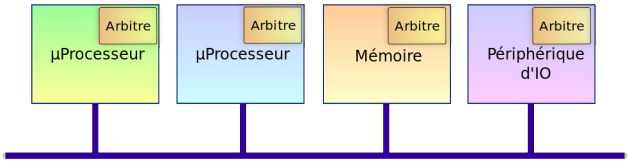
Arbitrage distribué



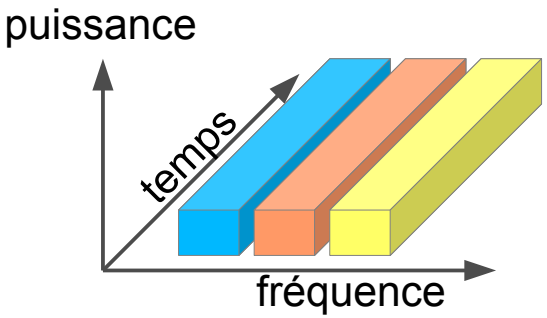
- FH-CDMA
  - Frequency-Hopping Code Division Multiple Access
  - principe : changer en permanence la fréquence d'émission
    - résistant aux interférences à bande étroite
    - difficile à intercepter

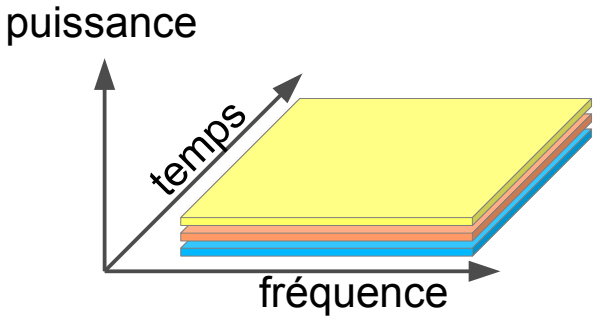


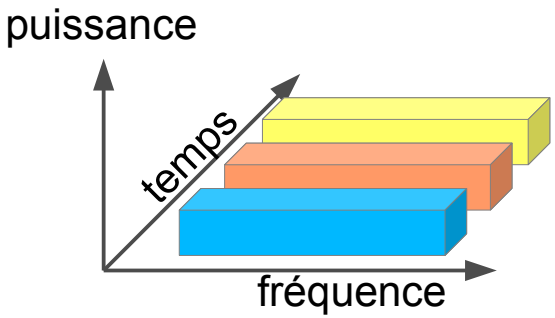
Arbitrage distribué



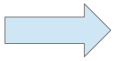
- FDMA / TDMA / FH-TDMA
  - FDMA : chacun sa fréquence
  - TDMA : chacun son tour
  - FH-TDMA / DS-CDMA : chacun sa séquence







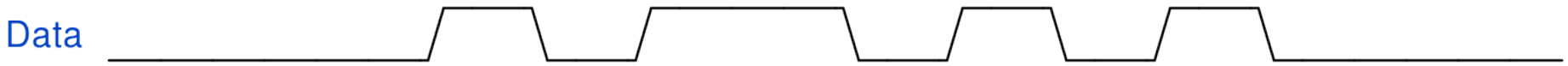
Plan



- Définition
- Structure physique
- Topologies
- Protocoles
- Arbitrage
- Timings
- Augmentation des performances
- Bus embarqués usuels

Timings

- **Quizz**
  - quels sont les bits transmis dans la séquence ci-dessous ?



Timings

- Il faut connaître la durée de chaque bit :
  - nécessite une convention sur la durée de bit
- plus généralement :
  - à tout moment, un signal a une *valeur*
    - 0 ou 1 (ou Z) pour un bit
    - n pour un bus
  - dans le temps, un signal est une suite d'*événements*
    - par exemple : 00010110101000
    - un événements est un *couple (valeur, date)*
  - **il faut pouvoir transmettre (encoder) un événement même si un signal ne change pas de valeur**

Timings

- **Encodage des événements**
  - encodage *périodique* : la date des événements est implicite
    - on définit au préalable la durée d'un événement (baud rate)
    - on définit un événement de référence (start bit, SOF, …)
    - les dates des autres événements s'en déduisent
  - encodage *apériodique* : la date des événements est explicite
    - sur papier, symboles séparés par des espaces, virgules, …
    - sur conducteur, une transition par événement
      - soit du signal lui-même (RZ, Manchester, …)
      - soit d'un signal de contrôle associé (horloge, strobe, …)

Timings

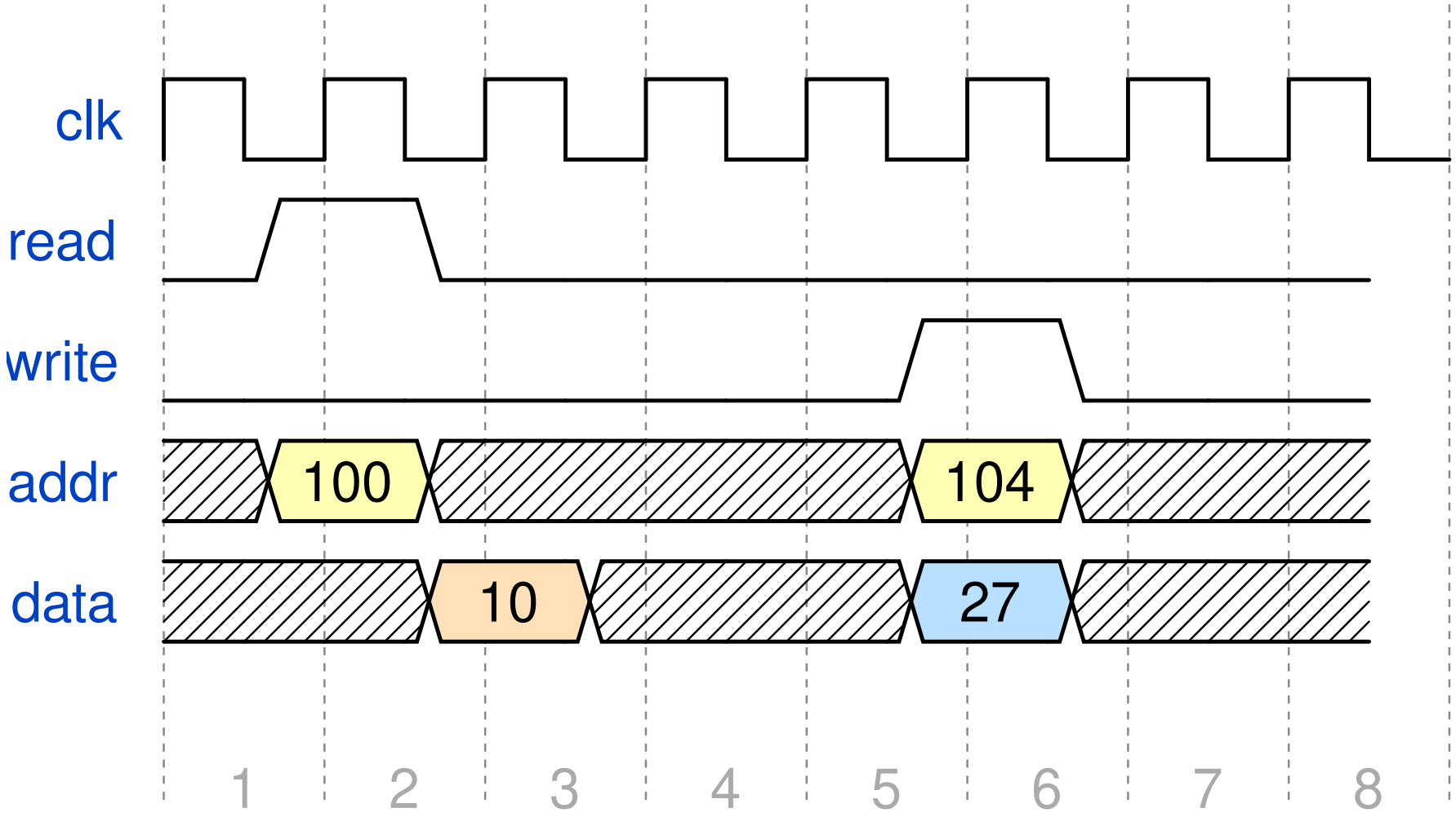
- **Finalisation de l'échange**
  - il faut garantir suffisamment de temps au récepteur pour lire l'événement transmis
    - garantir par construction une durée minimale du bit
    - utiliser un signal d'acquittement

Timings

- **Bus synchrone**
  - principe : un *événement* ne peut arriver qu'à des instants précis, fixés par une référence de temps commune à tous les nœuds appelée *horloge*.
  - des nœuds ayant une horloge commune définissent un *domaine d'horloge*
  - Remarques :
    - il faut transmettre la référence : ligne d'horloge
    - le cycle d'horloge devient la référence temporelle du protocole
    - il est possible de passer d'un domaine d'horloge à un autre, mais c'est délicat ("*synchronizer*")
    - bus locaux souvent synchrones

Timings

- **Bus synchrone**



Timings

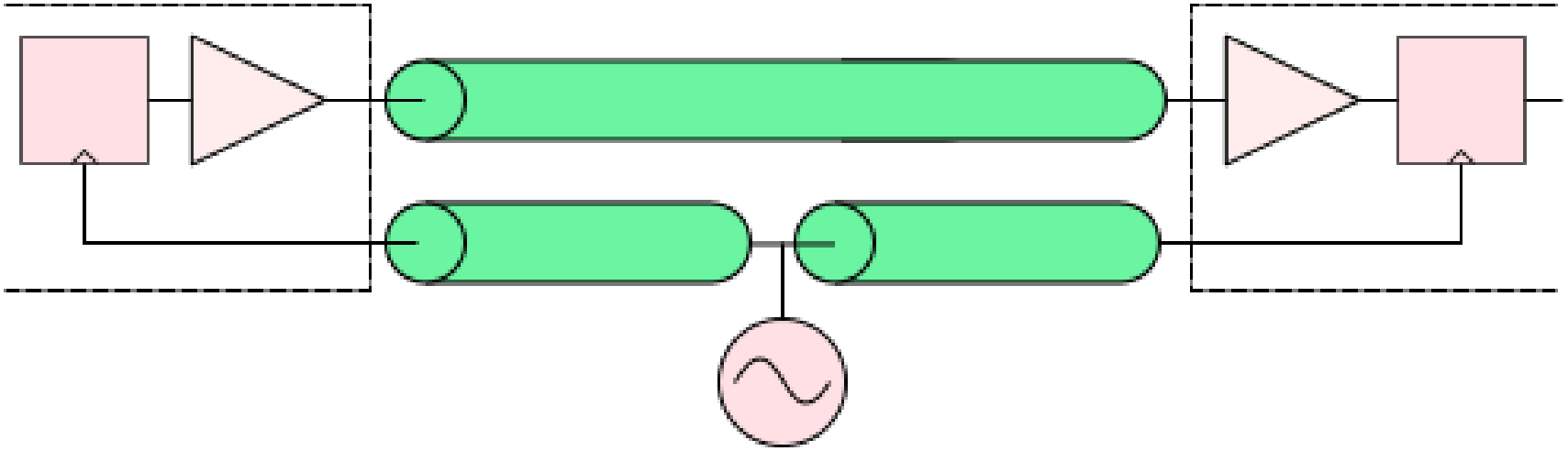
- **Bus synchrone**
  - Avantages
    - faible latence
    - peut fonctionner très vite
    - simple à implémenter
  - Inconvénients
    - tous les nœuds doivent fonctionner à la même vitesse
    - compromis vitesse / longueur du bus
  - une horloge strictement commune est impossible
    - différents types de synchronismes

Timings

- **synchrone / asynchrone**
  - mythe : synchrone / asynchrone
  - en réalité : plusieurs degrés de (a)synchronisme
    - synchrone
    - mésochrone
    - source-synchrone
    - plésiochrone
    - asynchrone

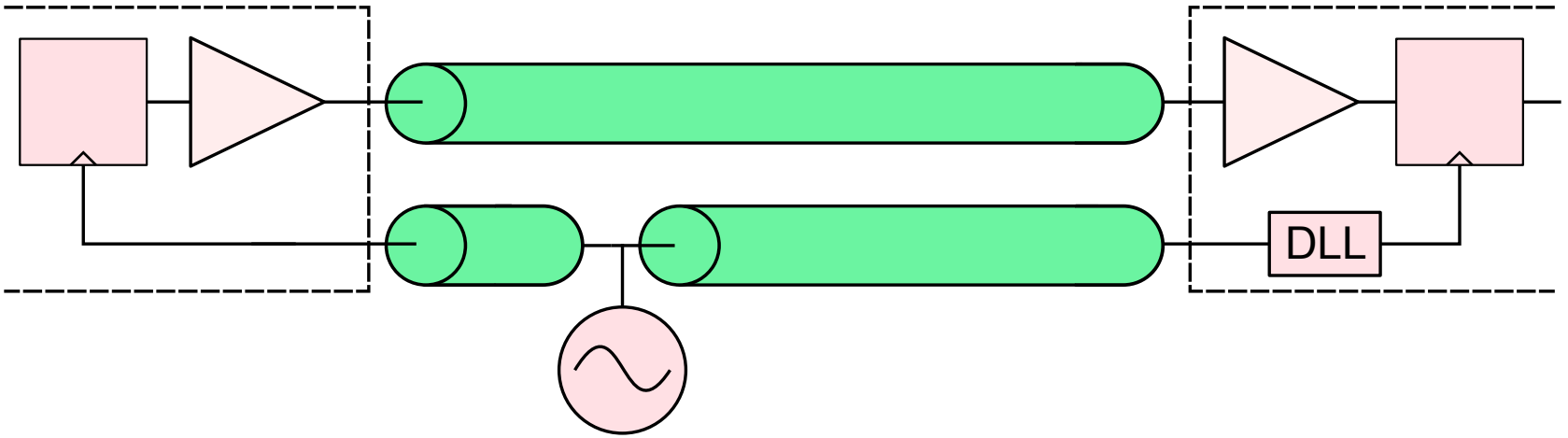
Timings

- **(a)synchronismes**
  - isochrone
    - même fréquence
    - même phase



Timings

- **(a)synchronismes**
  - mésochrone
    - même fréquence
    - phase différente
      - nécessite un moyen de corriger la phase à l'arrivée

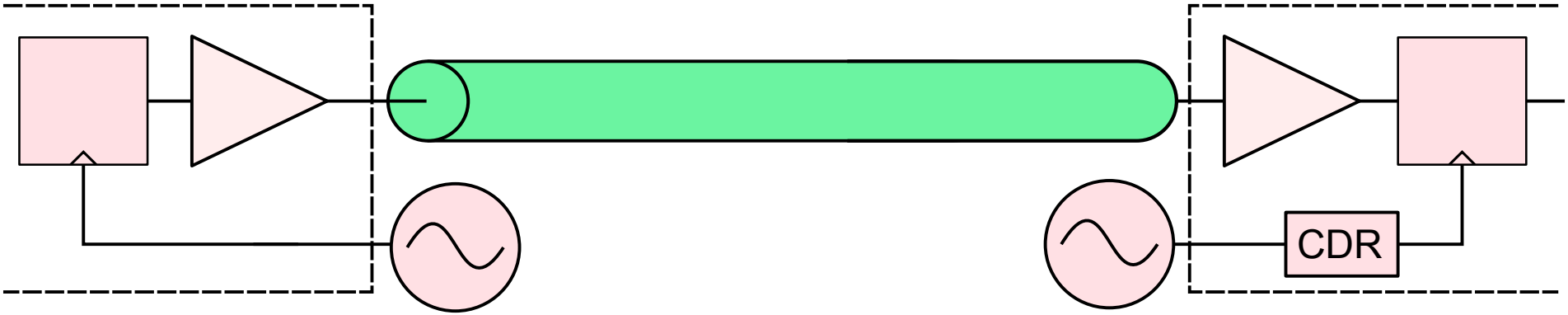


Timings

- **(a)synchronismes**
  - source-synchrone
    - l'horloge est transmise comme une donnée
    - fréquence limitée par les incertitudes seulement
      - skew
      - jitter

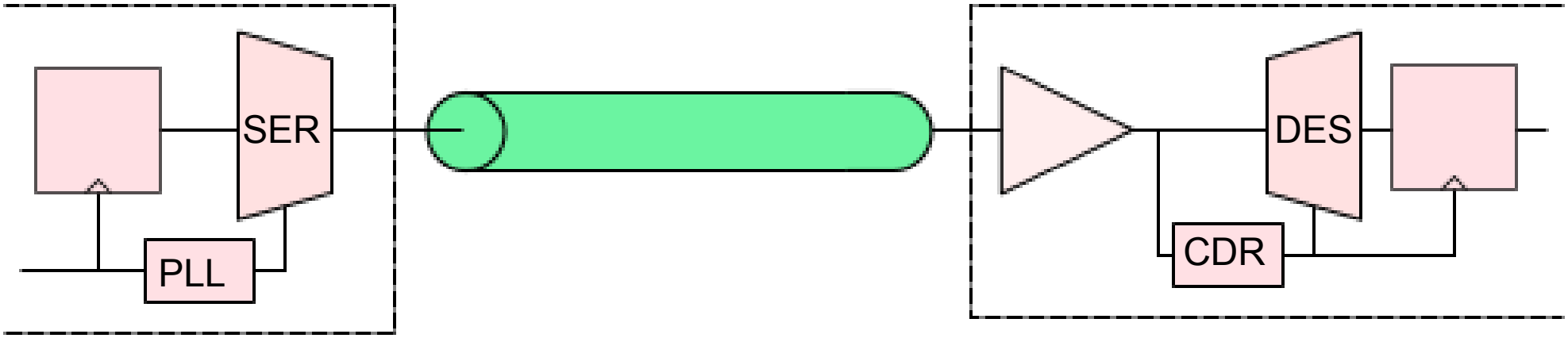
Timings

- **(a)synchronismes**
  - plésiochrone
    - fréquence légèrement différente
    - phase variant lentement
      - nécessite un moyen de corriger la fréquence et/ou la phase à l'arrivée : clock / data recovery (CDR)



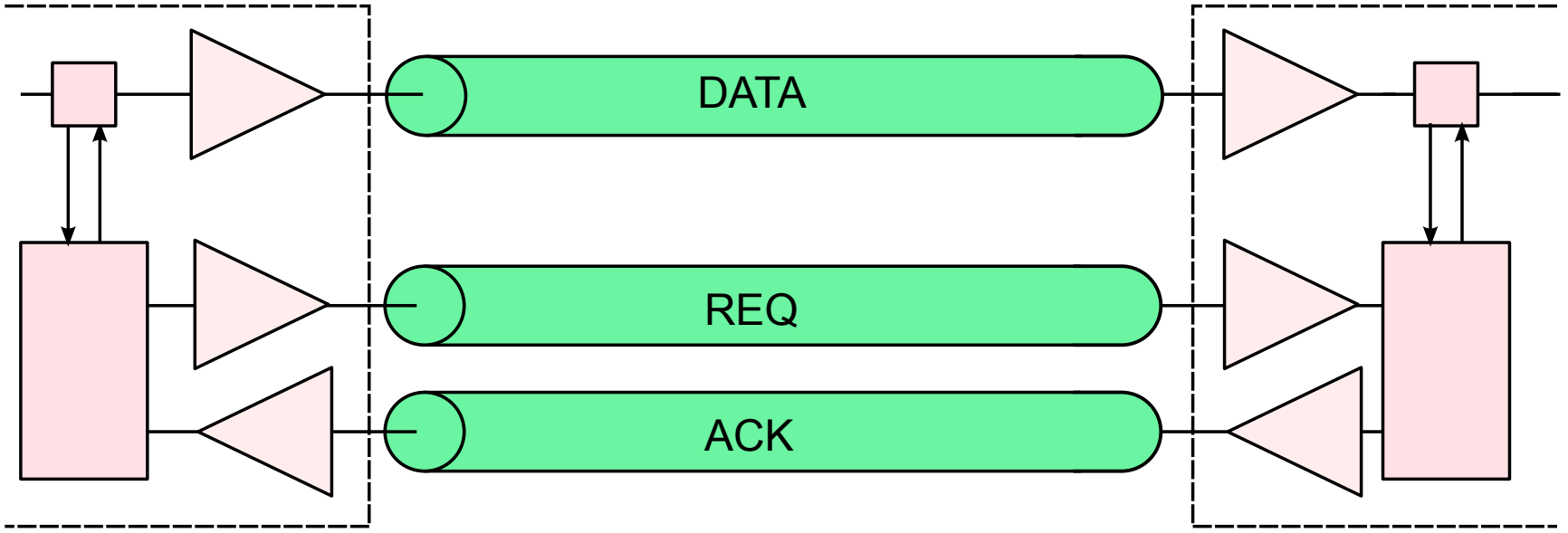
Timings

- **(a)synchronismes**
  - plésiochrone, exemple fréquent : clock-embedded
    - l'horloge est encodée dans le flux des données
    - nécessite un CDR
    - souvent utilisé avec un SERDES



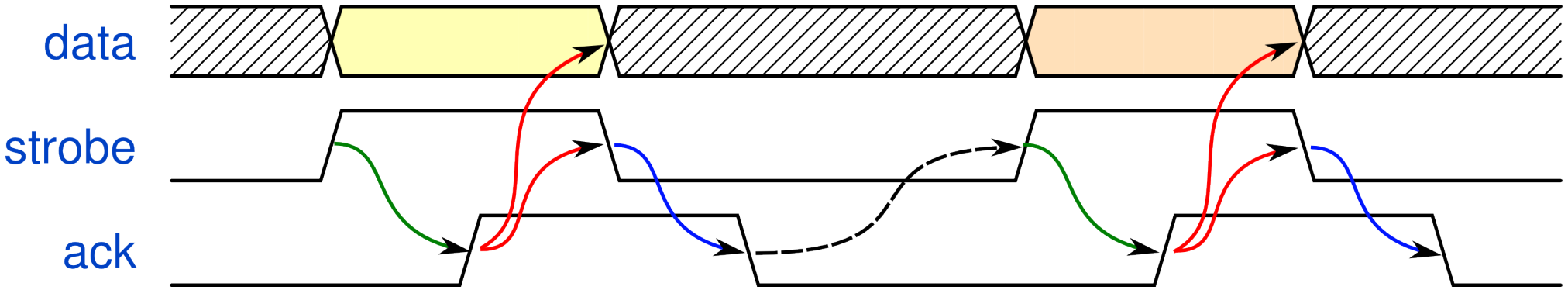
Timings

- **(a)synchronismes**
  - asynchrone avec handshake
    - pas d'horloge transmise
    - signaux de synchronisation pour assurer la transmission
      - souvent 4 phases
      - req / ack
      - strobe / rdy



Timings

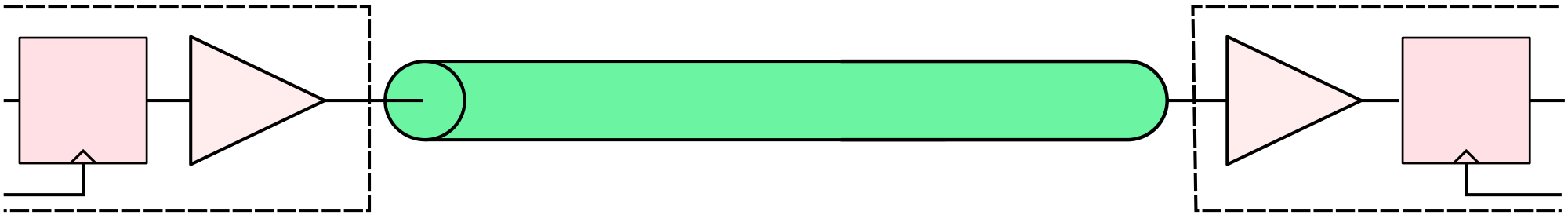
- **Handshake 4 phases**
  - exemple : strobe / ack



  - autres exemples :
    - req / ack
    - une phase + durée
    - …

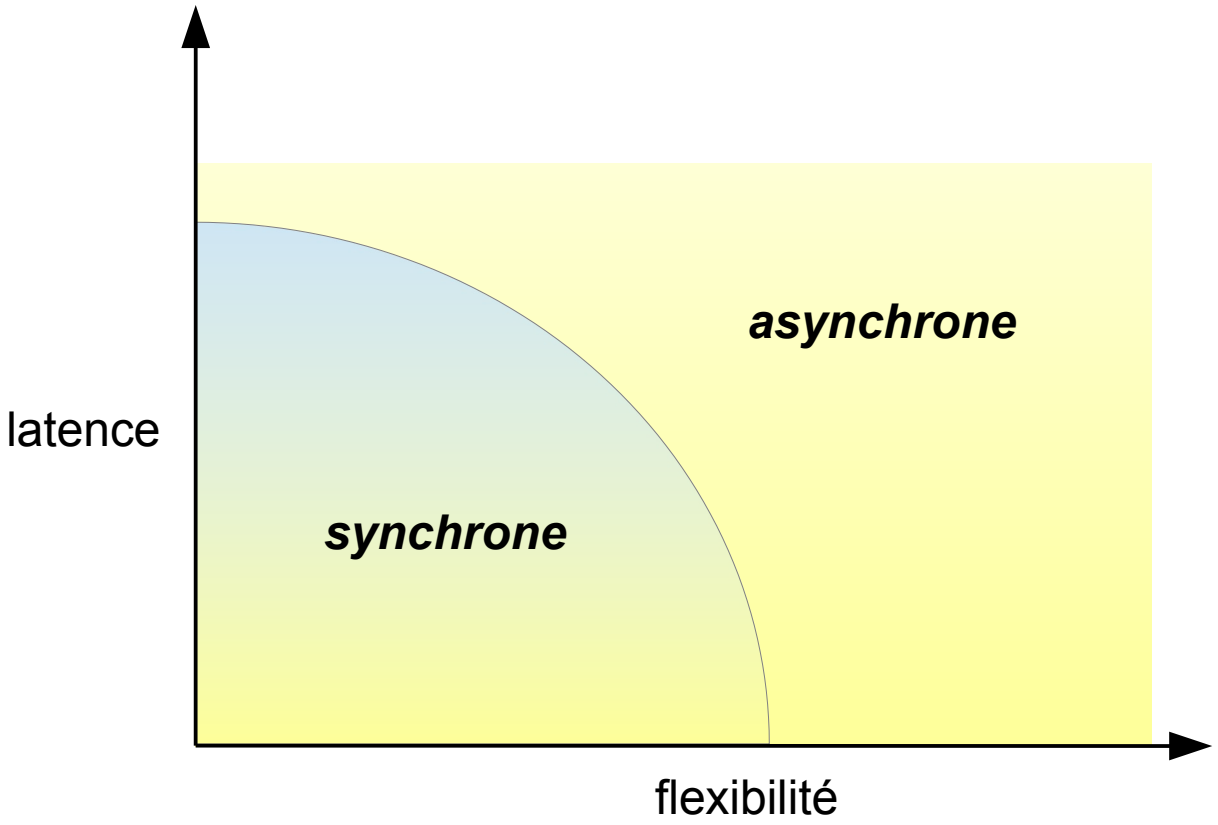
Timings

- **(a)synchronismes**
  - asynchrone sans handshake
    - pas d'horloge transmise
    - il faut se mettre d'accord sur la durée du bit
    - nécessité d'un bit de start
      - exemple : RS232, CAN
    - on peut prévoir un mécanisme de resynchronisation
      - exemple : CAN



Timings

- **Bus synchrone ou asynchrone ?**



Plan

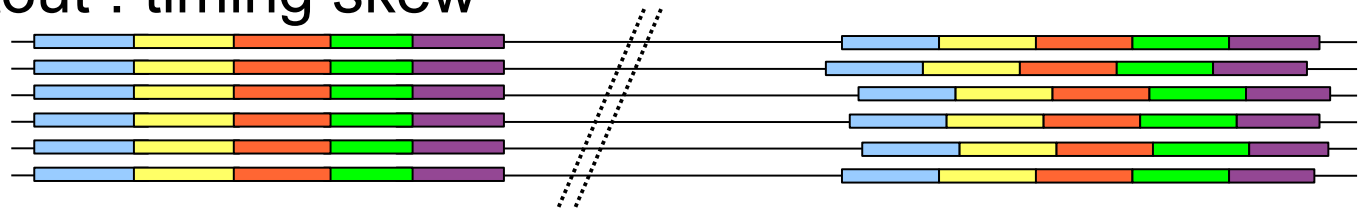
- Définition
- Structure physique
- Topologies
- Protocoles
- Arbitrage
- Timings
- Augmentation des performances
- Bus embarqués usuels

Bus parallèle / série

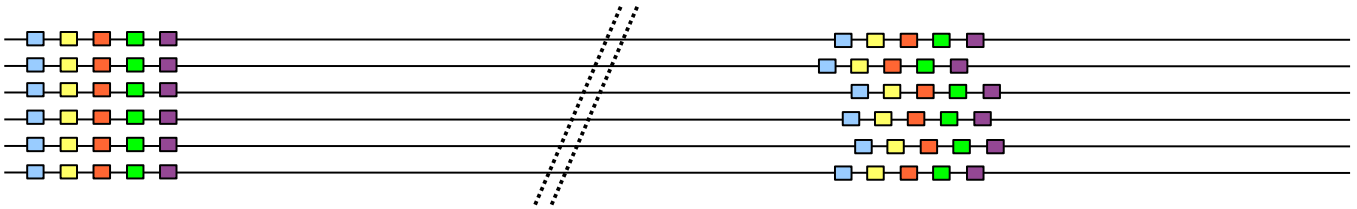
- **Bus parallèle**
  - plusieurs lignes de données
    - un bit par ligne
  - plusieurs lignes d'adresse
  - élargissement des mots : augmentation du débit
  - avantages
    - très facile à implémenter

Bus parallèle / série

- **Bus parallèle**
  - inconvénients
    - consommation
    - crosstalk
    - encombrement
      - routage
      - connecteurs / câbles
    - surtout : timing skew



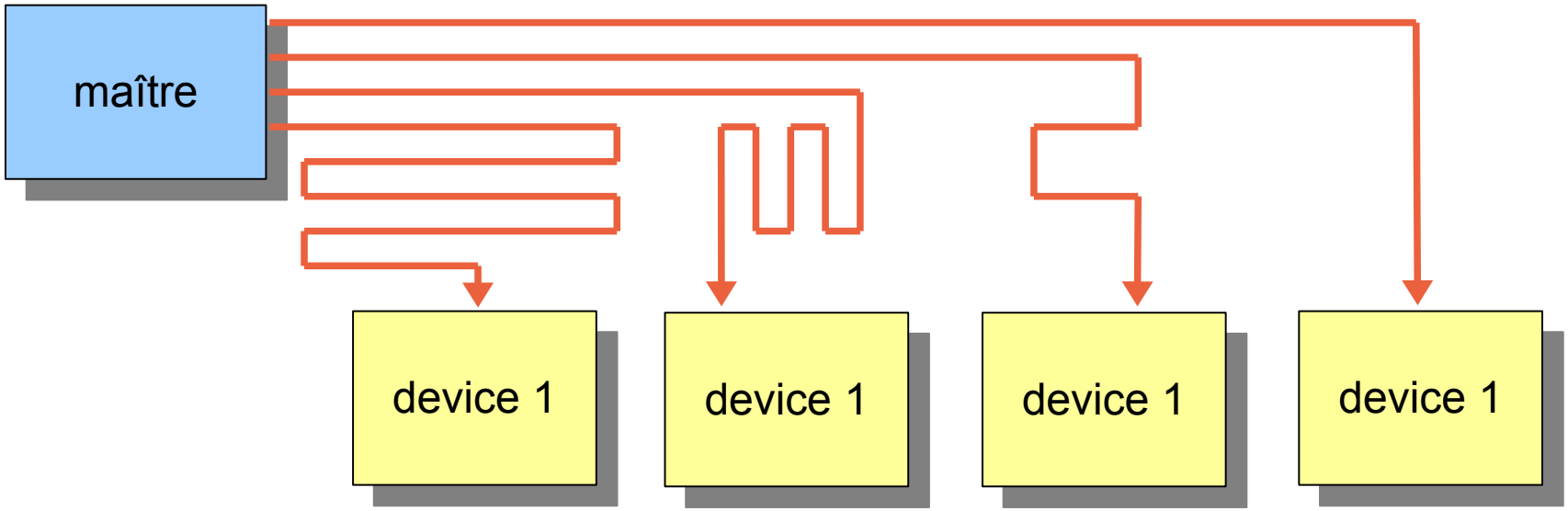
basse vitesse



haute vitesse

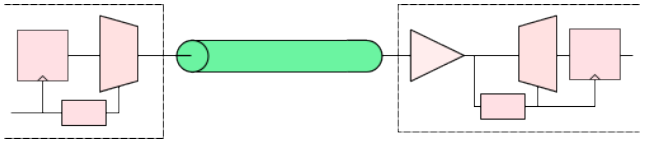
Bus parallèle / série

- **Bus parallèle**
  - réduction du timing skew
    - équilibrage des chemins



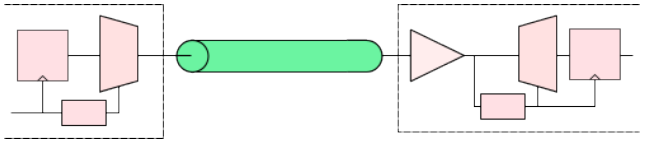
    - difficultés de routage
    - vitesse limitée par la longueur du chemin au dispositif le plus éloigné

Bus parallèle / série



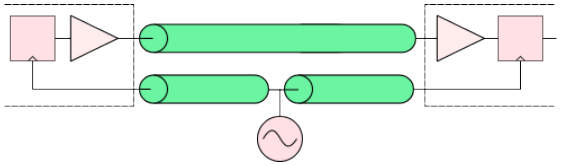
- **Bus série**
  - un unique fil de données
  - peut aussi transmettre le contrôle et/ou l'alimentation
  - les mots sont transmis bit par bit
  - paradoxalement : permet d'augmenter le débit, surtout aux longues distances
    - moins de fan out
    - moins de skew
    - donc fréquence de fonctionnement potentiellement très élevée
    - surtout si source-synchrone et pas d'acquittement

Bus parallèle / série



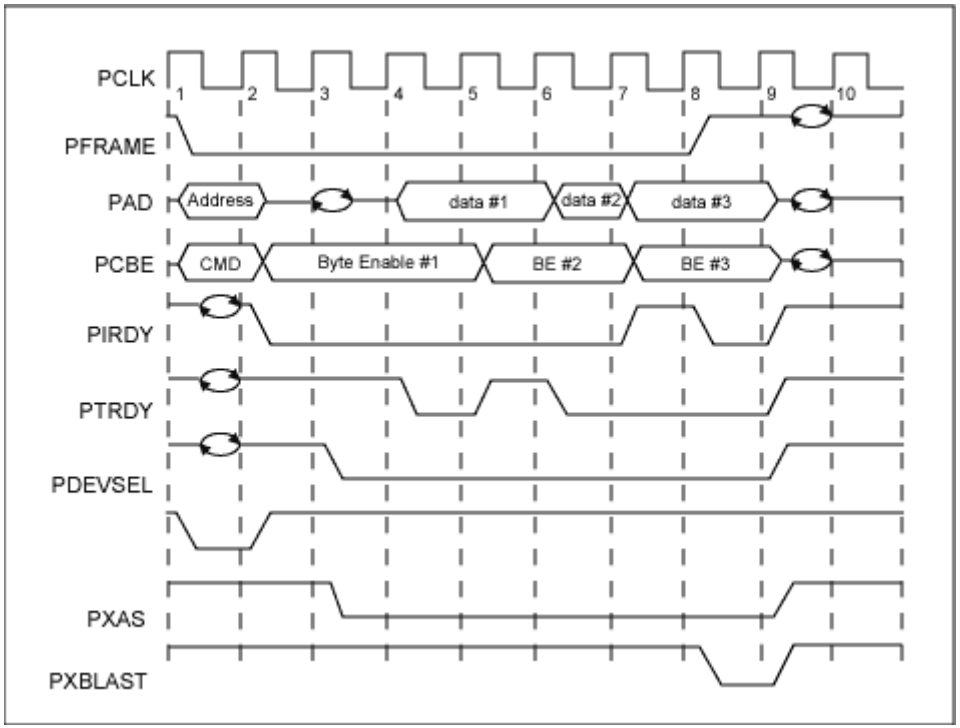
- **Bus série**
  - avantages :
    - moins cher, moins encombrant
  - inconvénient : plus complexe à implémenter
    - l'émetteur doit sérialiser les mots
    - le récepteur doit les dé-sérialiser
    - si le contrôle passe sur la même ligne, le protocole devient très complexe

Bus parallèle / série

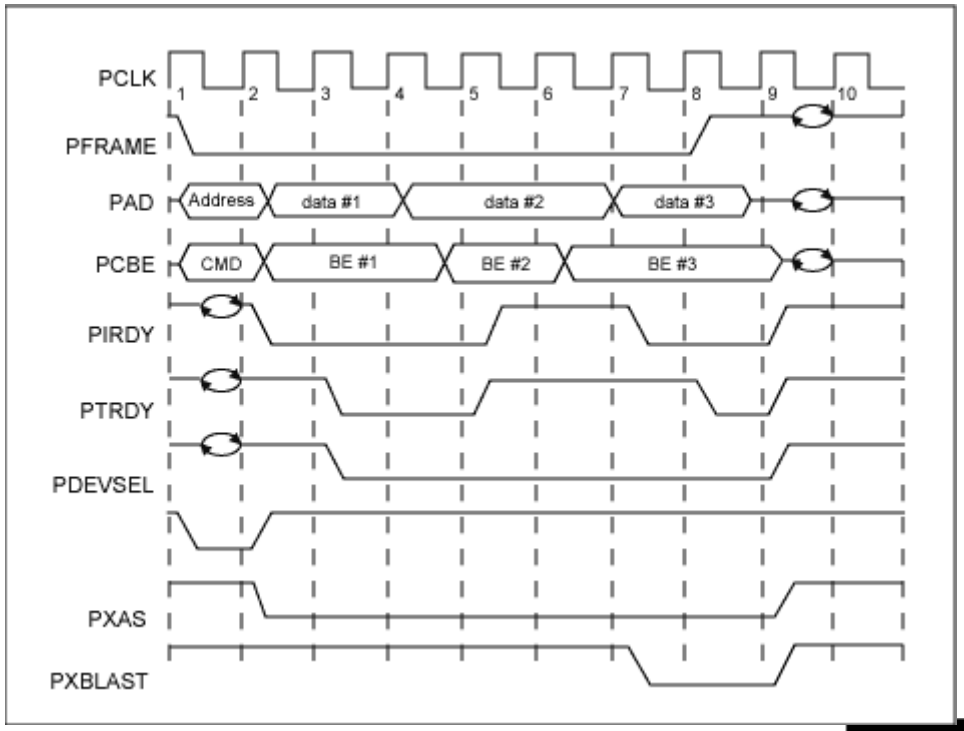


- **Bus multiplexé**
  - mi-chemin entre série et parallèle
  - exemple : bus PCI

PCI read



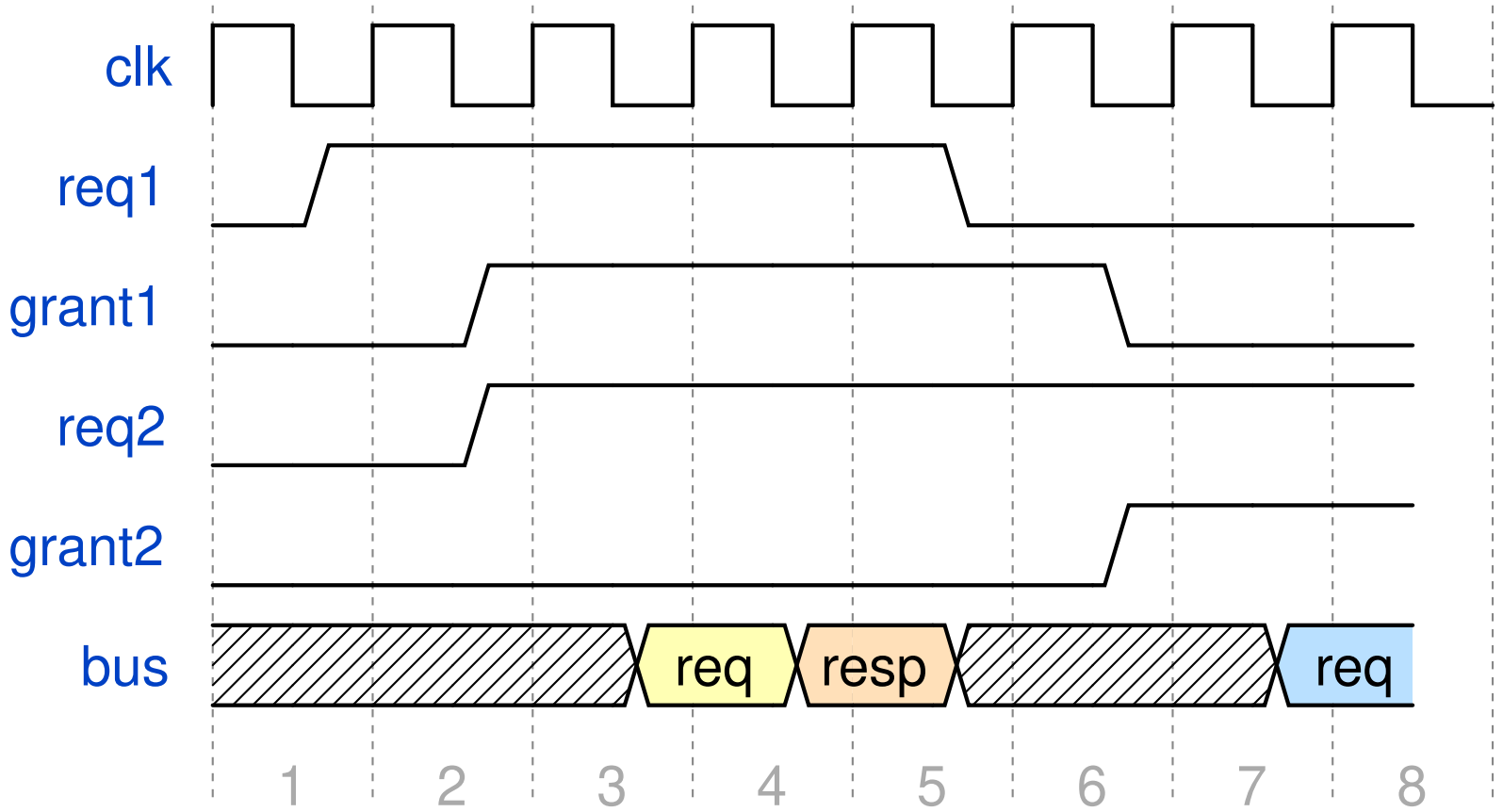
PCI write



source : Maxim IC

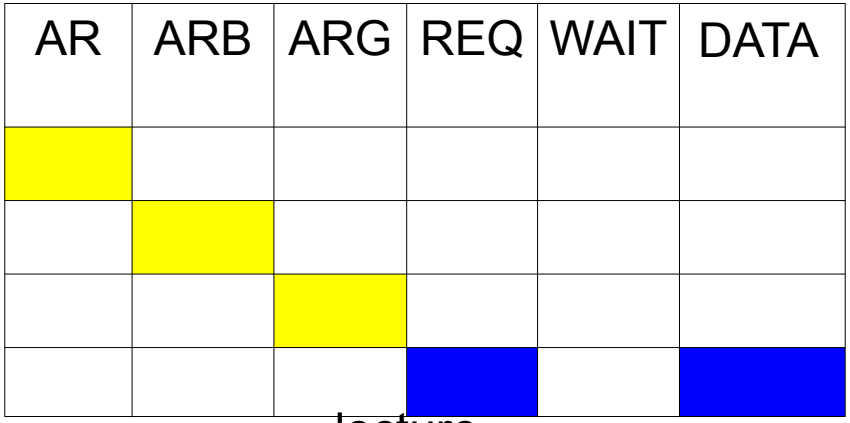
Augmentation des performances

- **Protocole basique d'arbitrage synchrone**



Augmentation des performances

- **Pipe-line des accès**
  - différentes phases peuvent souvent se chevaucher
    - arbitrage / transfert
    - pause / transfert
  - Exemple sans pipe-line :



lecture

écriture

Augmentation des performances

- **Pipe-line des accès**
  - utilisation plus efficace du bus
    - temps d'attente
    - arbitrage overlap

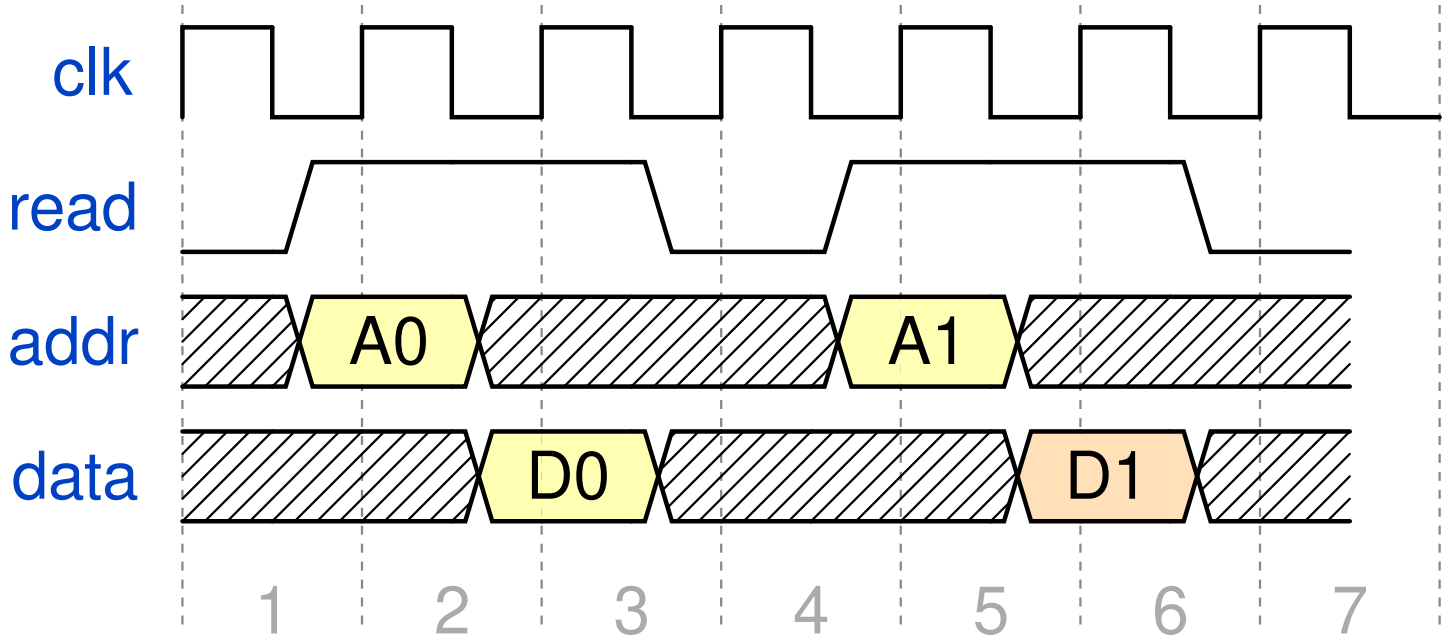
read	AR	ARB	AG	REQ	WAIT	DATA									
write		AR	ARB	AG	-	-	REQ	DATA							
write			AR	ARB	ARB	ARB	AG	-	REQ	DATA					
read				AR	ARB	ARB	ARB	ARB	AG	-	REQ	WAIT	DATA		
read							AR	ARB	ARB	ARB	AG	REQ	WAIT	DATA	
read								-	AR	ARB	ARB	AG	-	-	REQ

Augmentation des performances

- **Split-transaction**
  - les transferts s'effectuent en deux phases décorrélées
    - envoie d'une commande
    - réception de la réponse
  - surtout utile si attente variable dans les transactions
  - pour chacune des phase, le nœud participe à une phase d'arbitrage
  - nécessite une notion de Request ID
  - atomicité (cohérence) : possibilité de poser des locks

Augmentation des performances

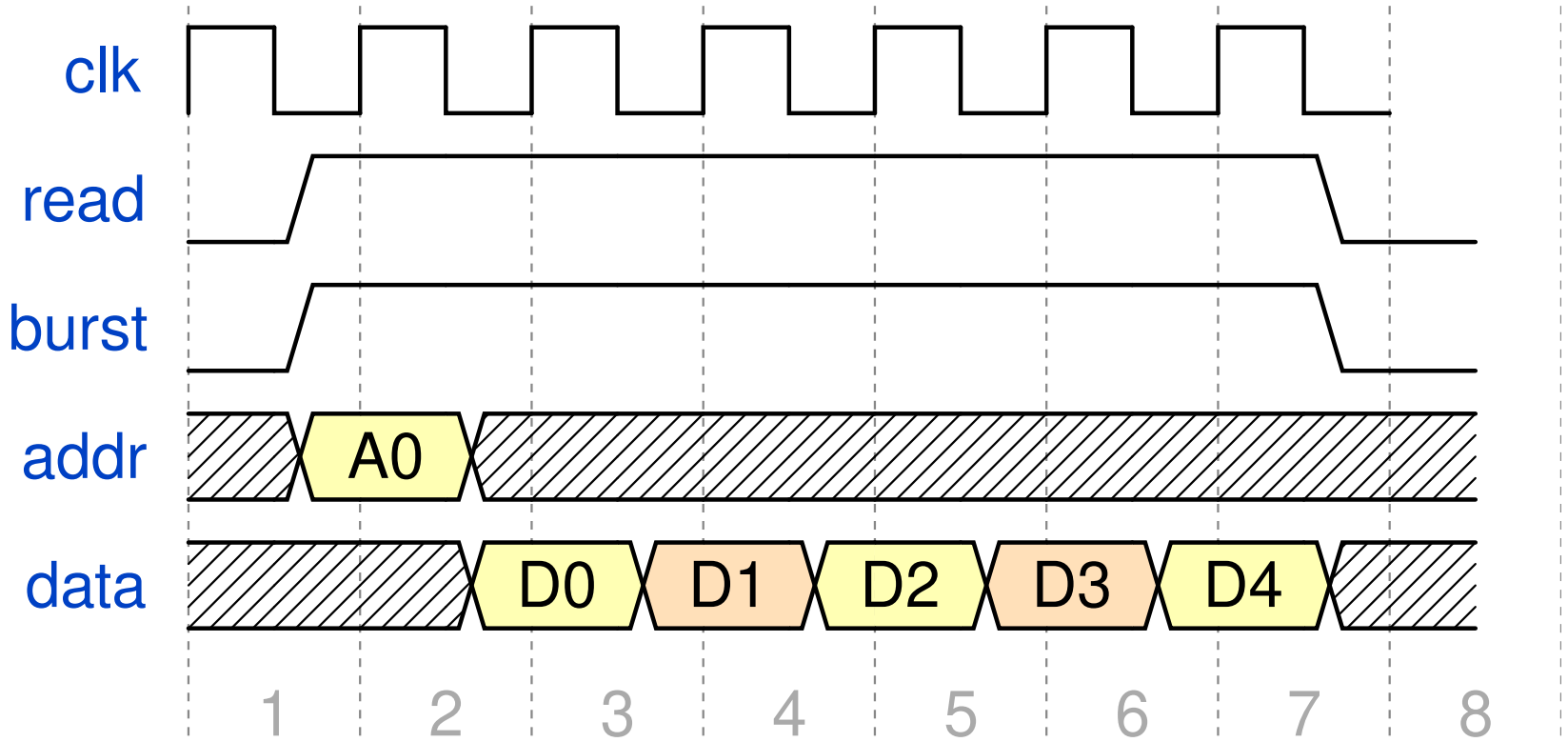
- **Burst**
  - un accès normal a un overhead énorme
    - arbitrage
    - envoi de la commande
    - acquittement



ici : efficacité = 33%…

Augmentation des performances

- **Burst**
  - tirer partie de la localité des accès
  - adresse de départ, puis données les unes à la suite des autres



Augmentation des performances

- **Burst**
  - compteur d'adresse :
    - linéaire
    - entrelacé
  - burst long :
    - augmente l'efficacité
    - augmente la latence des autres nœuds
  - solutions possibles :
    - longueur maximum
    - interruption des burst puis restart ou resume

Plan

- Définition
- Structure physique
- Topologies
- Protocoles
- Arbitrage
- Timings
- Augmentation des performances
- Bus embarqués usuels

Bus courants dans l'embarqué

- RS232 :
  - bus lent (9600 – 115200 bps), bidirectionnel, point à point
  - usuel sur les vieux PC, courant dans les systèmes embarqués
- RS422, RS485 :
  - différentiel, half-duplex, rapides, multidrop
- I2C :
  - bus lent (400kbps), multidrop, multi-maîtres, arbitrage distribué
  - adressage simple
  - variantes : SMBUS, DDC, TWI, …
- CAN :
  - bus semi-rapide (1Mbps), multi-maîtres, arbitrage distribué automatique
  - pas d'adressage (mais message ID)
  - peu sensible au bruit
  - voir aussi : LIN, FlexRay

Bus courants dans l'embarqué

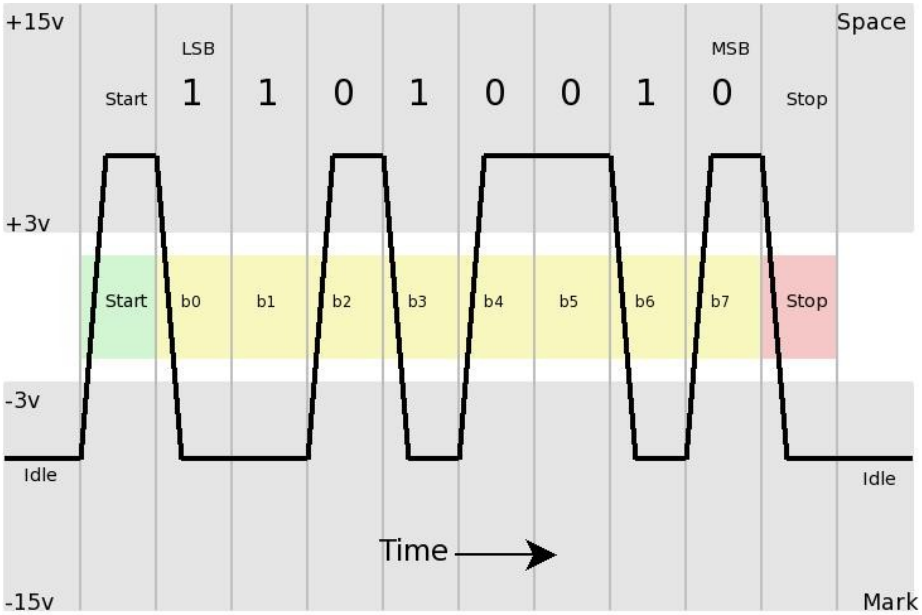
- SPI :
  - bidirectionnel (full-duplex), 10Mbps
  - un maître, plusieurs esclaves : mélange de multidrop et point à point, daisy-chain possible
  - extrêmement simple, utilisé pour flash, EEPROM, MMC / SD, Ethernet, son, LCD, capteurs, ...
  - variantes : I2S

- USB :
  - bus multi-vitesse (LS, FS, HS), différentiel, topologie arbre
  - 1 maître, plusieurs esclaves
  - courant dans les PC

- Firewire / IEEE1394 :
  - haut débit (100-800 Mbps), isochrone, topologie arbre
  - multi-maîtres, point à point
  - plus efficace que l'USB, mappé en mémoire

RS232

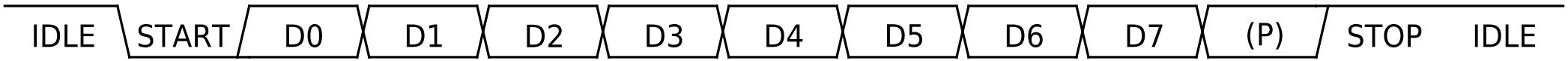
- Couramment appelé « port série »
- Électriquement
  - asynchrone (9600, 19200, 38400, 57600, 115200 bps)
  - 1 (repos, mark) : -3 à -15V, 0 (actif, space) : +3 à +15V
  - 2 fils : RX / TX
- Protocole
  - point à point, full duplex
  - contrôle de flux logiciel (échappement) ou matériel (fils supplémentaires)
  - format des données
    - data : 5, 6, 7, 8, 9 bits
    - parité : 0 ou 1 bit
    - stop : 1, 1.5, 2 bits
  - Notation usuelle : 115200 8N1



wikipedia

RS232

- Au niveau logique
  - avant transceiver, en sortie du microcontrôleur :

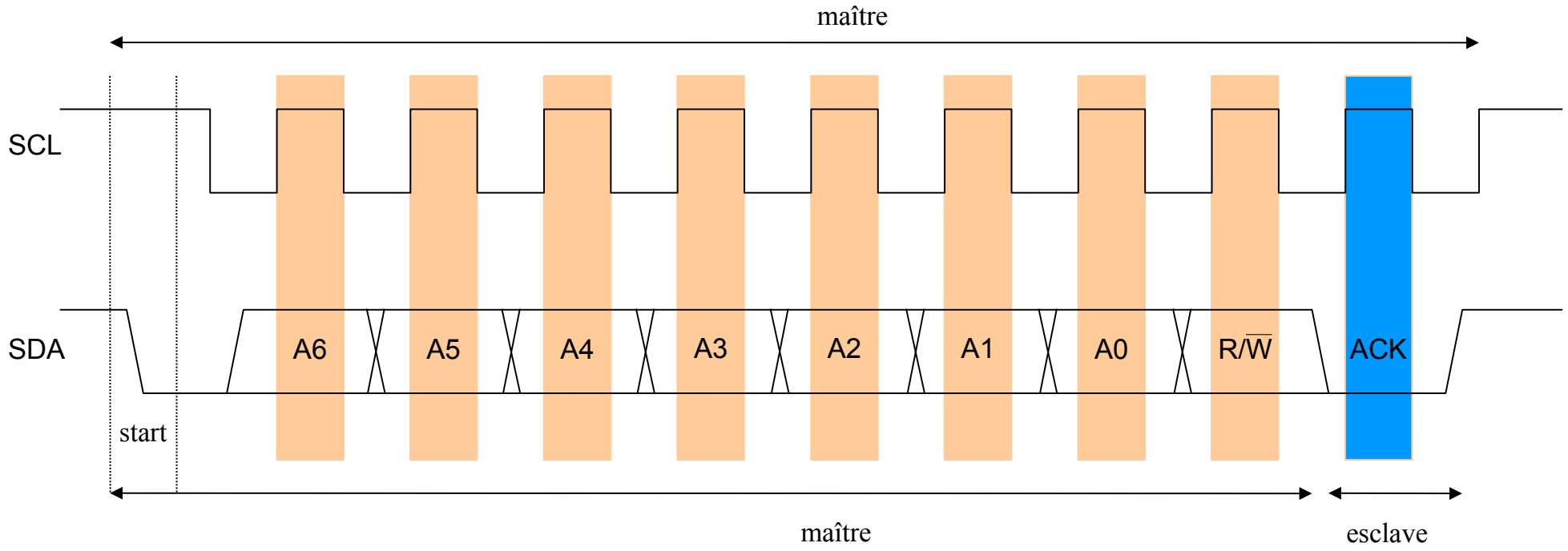


- Attention à la dérive en fréquence
  - Si on est en 8N1, quelle dérive maximum est admissible ?
  - Problème si le diviseur d'horloge ne tombe pas pile sur la bonne fréquence…
- Erreurs
  - Framing error : bit de stop non reconnu.
  - Overrun : buffer vidé trop tard.
  - Parity : erreur de parité.
- Attention au nommage des fils !

I2C

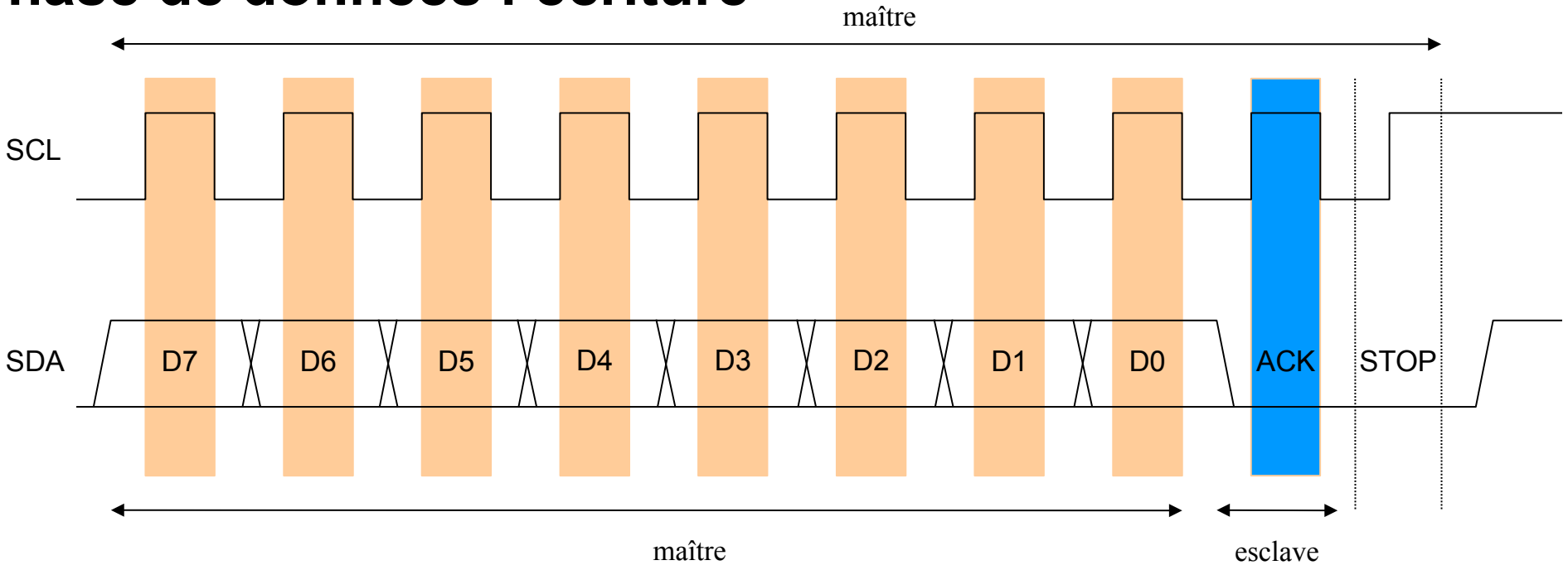
- Électriquement
  - 2 fils (SCL / SDA), collecteur ouvert (donc pull-up !)
  - 1 (repos) : 3.3 – 5V, 0 (actif) : 0V
- Protocole
  - Maître : impose l'horloge et transmet les adresses
    - transition sur SDA lorsque SCL = 0 : donnée
    - transition sur SDA lorsque SCL = 1 : start ou stop
  - Clock stretching : un esclave trop lent peut maintenir SCL bas.
  - Pas de sémantique sur les messages (à part adresse 0 : « general call »)
  - Message :
    - une phase d'adresse
    - une ou plusieurs phases de données
  - Arbitrage tout au long du dialogue
    - dominant : 0
    - récessif : 1

Phase d'adresse



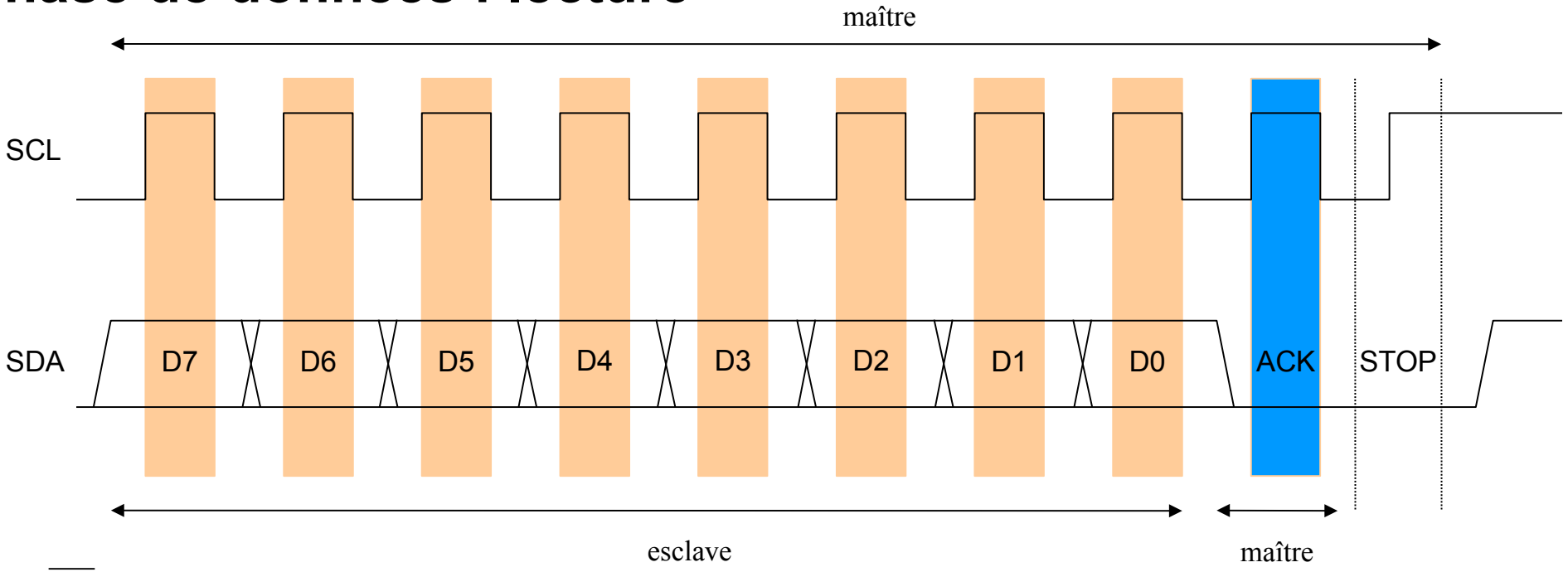
- Un esclave ayant reconnu son adresse répond par un ACK.
- Un peu moins de 128 adresses disponibles
- Adresses réservées :
  - 0000000 : general call (réglage adresses, SOS, ...)
  - 0000001 – 0000111, 111111xx : réservées
  - 11110xx : 10 bits

- **Phase de données : écriture**



- R$\overline{W}$ dans la phase d'adresse = 0.
- Le maître envoie 0 ou plusieurs octets à l'esclave qui répond par un ACK.
- La transaction se termine par un STOP.
- L'esclave interprète les données comme il le veut.

Phase de données : lecture



- $R\overline{W}$ dans la phase d'adresse = 1.
- L'esclave envoie un octet.
- Le maître envoie
  - un ACK pour continuer à lire d'autres octets,
  - un NACK sinon.
- La transaction se termine par un STOP.

I2C

- Écritures / lectures combinées
  - Un maître peut combiner des écriture et lectures sans passer par l'état STOP.
  - Exemple :
    - écriture : le maître spécifie à la RAM une adresse à lire,
    - lectures : le maître lit les données de la RAM.
- Un maître peut envoyer un STOP à n'importe quel moment.
- Utilisation courantes :
  - IO expander
  - flash, eeprom, RAM
  - DDC, SPD
  - DAC et ADC lents
  - contrôleurs vidéo, audio, …
- Variante : SMBUS

SPI

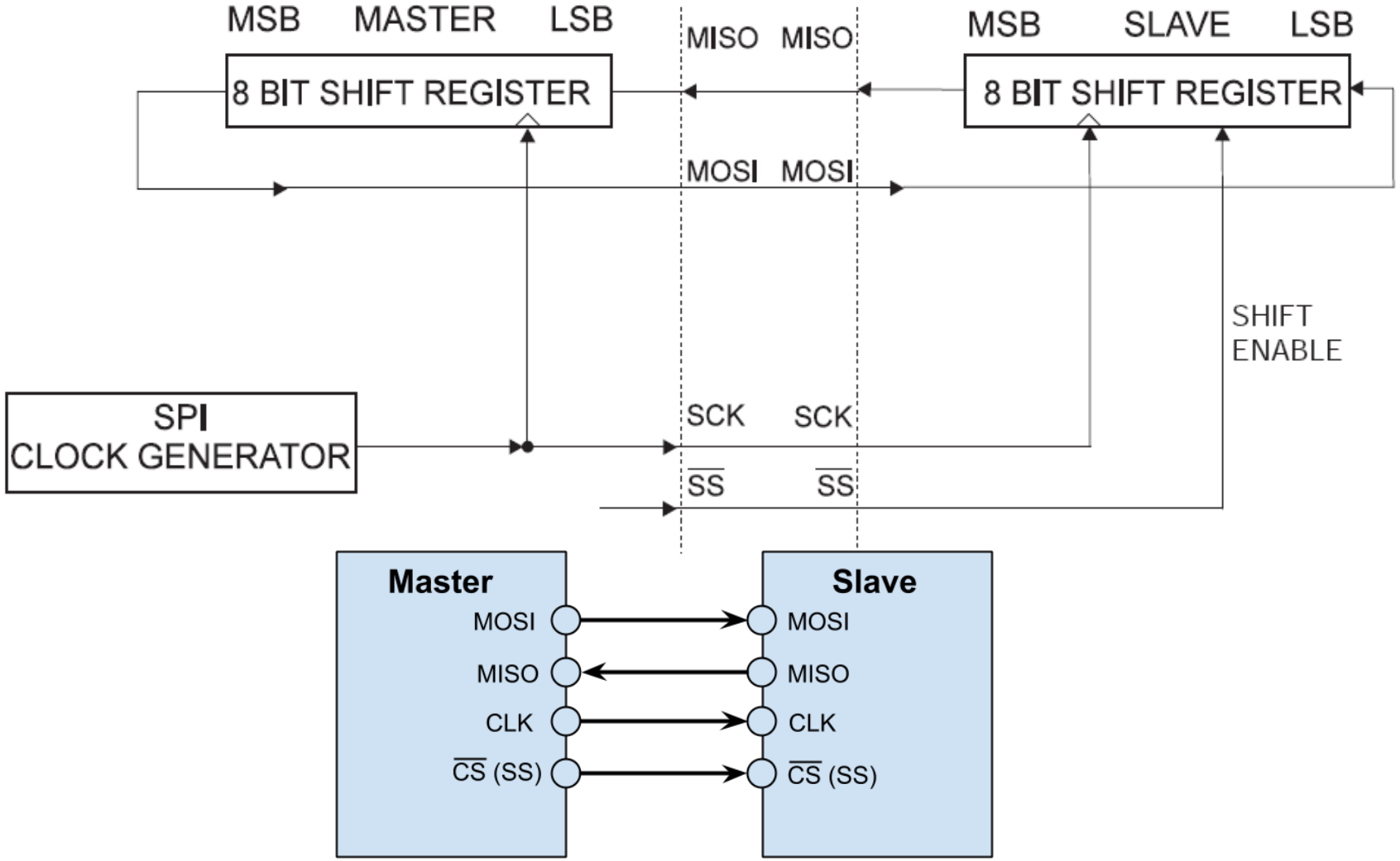
- Serial Péripheral Interface bus, utilisé pour :
  - capteurs : température, pression, ADC, DAC
  - communications : Ethernet, CAN, ...
  - flash et EEPROM
  - écrans LCD
  - cartes MMC et SD

- Électriquement
  - 3 ou 4 fils (CS, SCK, MISO, MOSI), point à point ou daisy chain.
  - logique CMOS ou TTL.
  - 10Mhz en standard. Peut parfois monter jusqu'à 40MHz.

- Protocole
  - Un maître, un ou plusieurs esclaves.
  - full-duplex :
    - MOSI : données transitant du maître vers l'esclave
    - MISO : données transitant de l'esclave vers le maître
    - SCK : horloge
    - CS : chip select

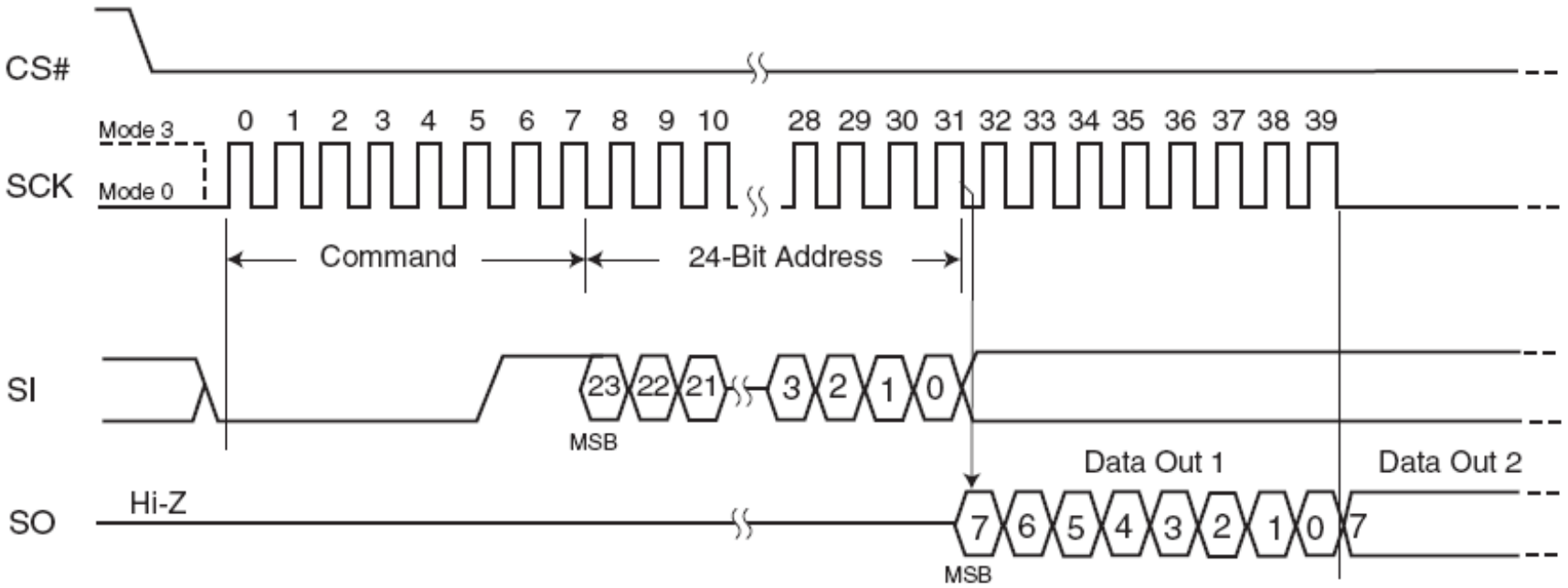
SPI

- Transmission
  - registres à décalage : échange des données



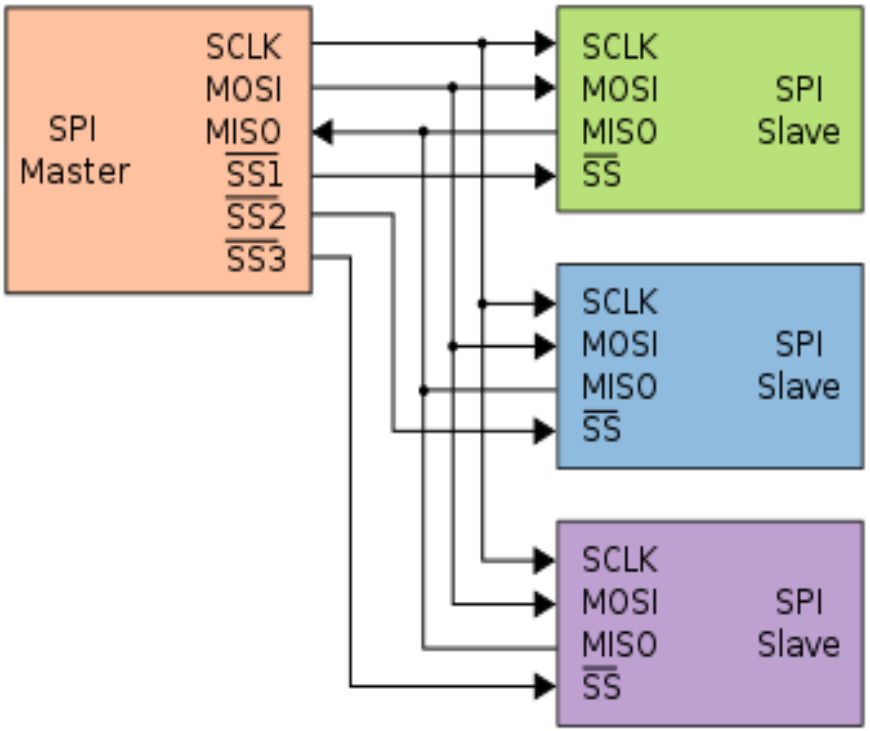
SPI

- autant de bits que nécessaire (usuellement 8) :
  - codec audio : 16 bits
  - DAC / ADC : 12 bits
  - flash : 8n+32

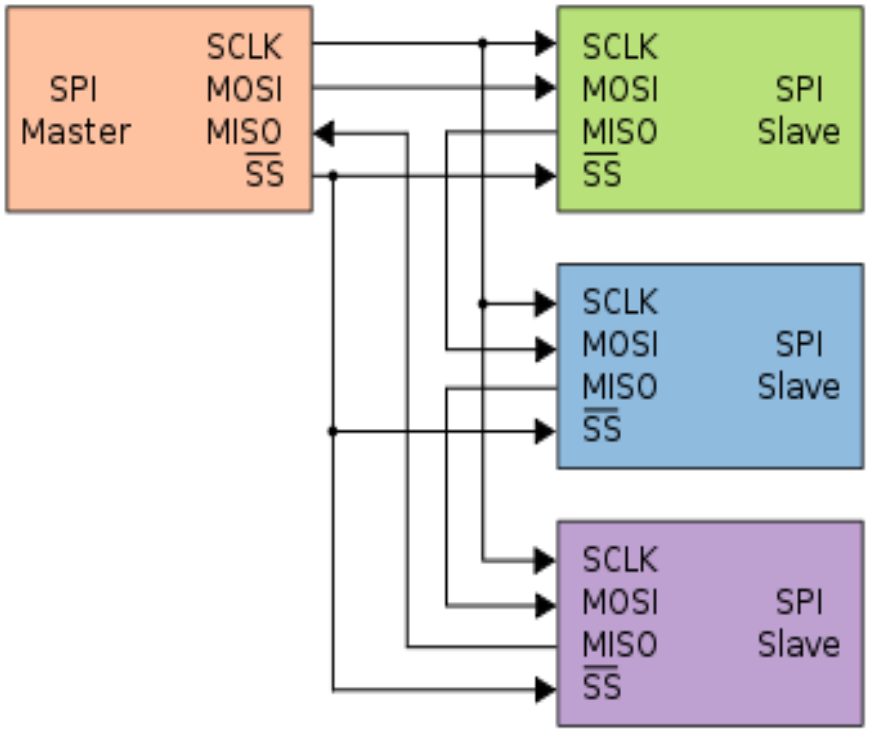


SPI

- Plusieurs esclaves
  - Point à point
  - Daisy chain

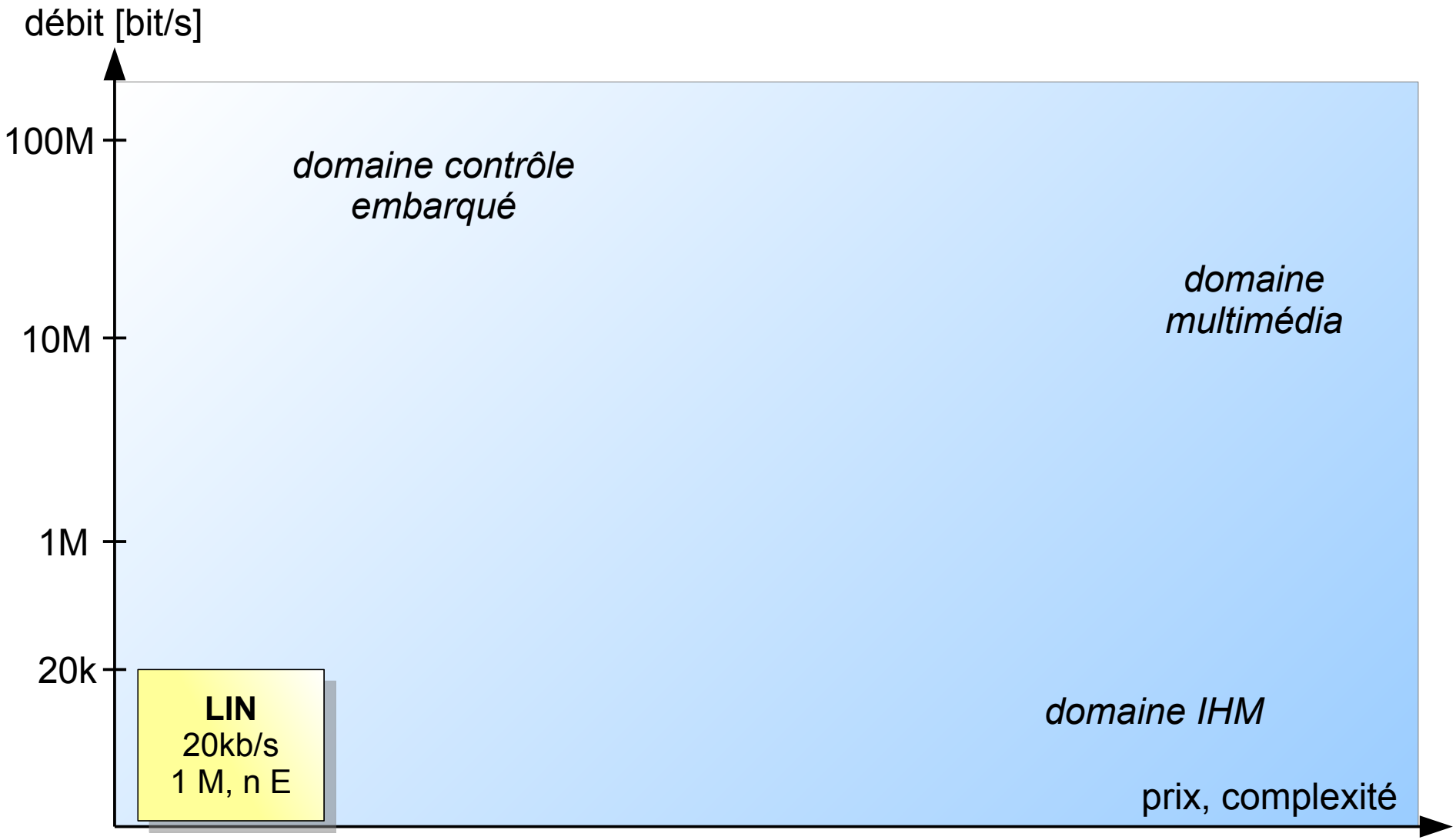


wikipedia



wikipedia

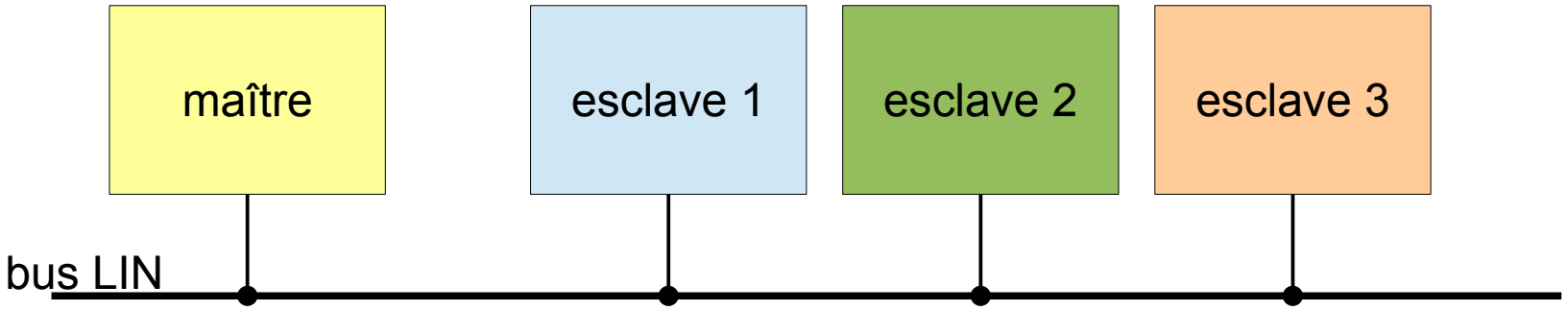
Bus automobiles



source : Freescale

LIN

- **Le bus LIN**
  - Bus automobile très bas coût, communications à très faible débit
    - fermeture des portes
    - lève-vitre
    - contrôle des miroirs
  - Débit : 20kb/s
  - Basé sur UART standard
    - 1 maître, plusieurs esclaves
    - 1 fil

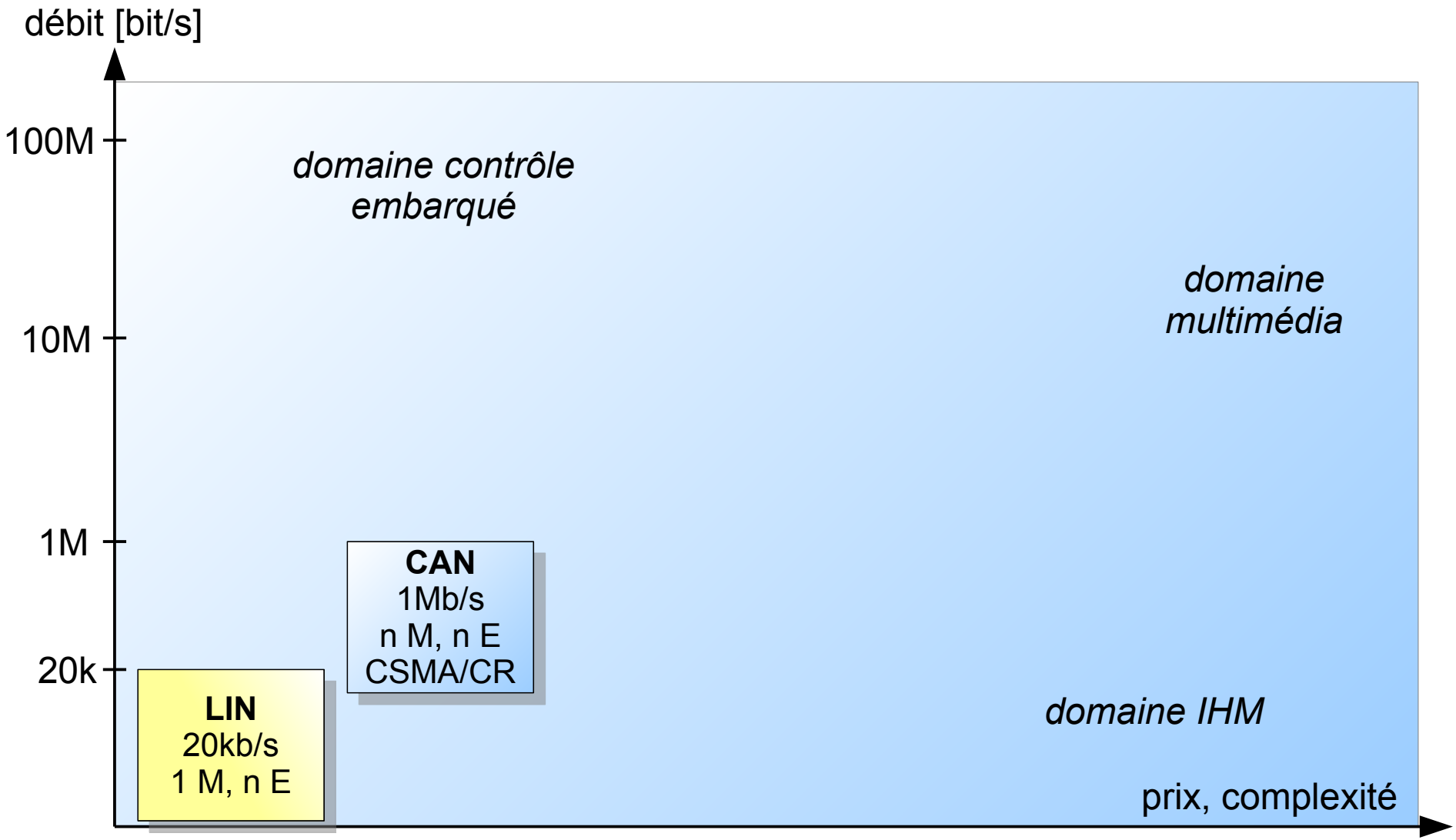


LIN

- **Le bus LIN**
  - Format des trames

  - Break (SOF) : 13 bits dominants + 1 bit récessif
  - Sync : 0x55 (pourquoi ?)
  - ID : 6 bits + 2 de parité
    - 0 – 59 : trames de données
    - 60, 61 : trame de diagnostic
    - 62, 63 : réservées
  - Data : 1 à 8 octets
  - Checksum

Bus automobiles



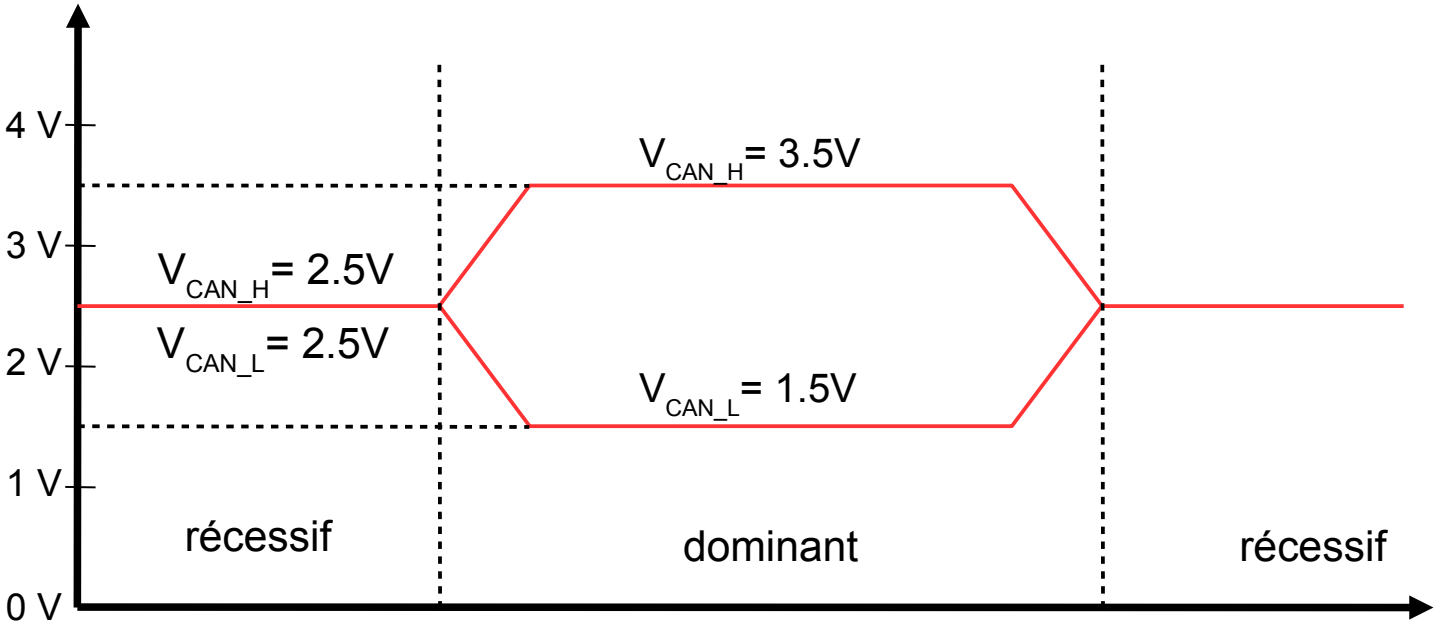
source : Freescale

CAN

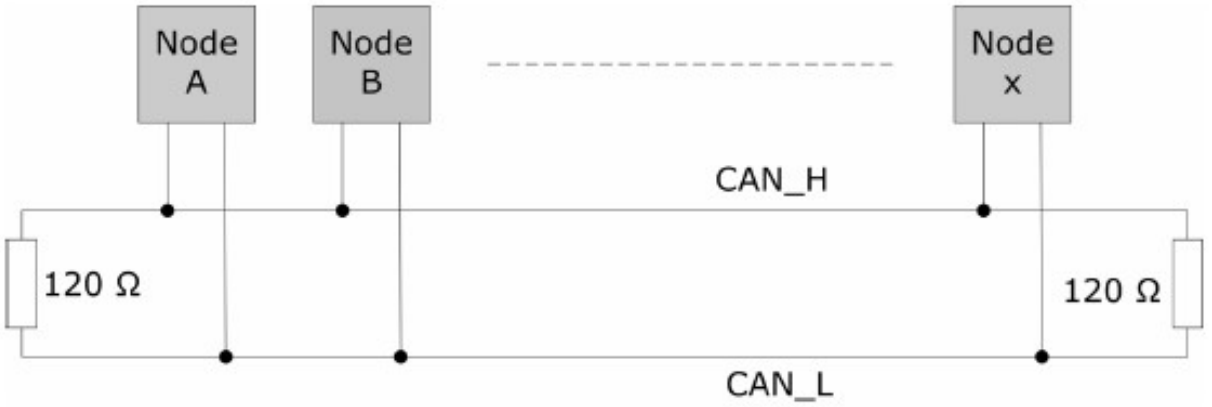
- Controller Area Network
  - très utilisé dans l'automobile et l'industrie
  - robuste, faible latence, temps-réel (CANTT)
  - 2014 : plusieurs milliards de nœuds CAN dans le monde
- Électriquement:
  - bus différentiel
  - vitesse variable : 1Mbps (40m) à 10kbps (5km)
  - multidrop
- Protocole :
  - multi-maîtres, multi-esclaves
  - arbitrage distribué
  - broadcast : pas d'adressage (content-adressed)
  - messages courts : 8 octets maximum
  - détection et confinement des erreurs

CAN

- **Signalisation**
  - Deux états
    - 0 - dominant : Vdiff > 2V
    - 1 - récessif, repos : Vdiff = 0V
    - électriquement équivalent à du collecteur ouvert (mais différentiel)
  - Nécessite un transceiver



- **Signalisation (suite)**
  - Topologie multipoint.
  - Le bus doit être terminé par des résistances de 120 ohms (cf cours d'intégrité du signal).



CAN

- **Messages**
  - 4 types de messages (trames)
    - Data frame : « voici le message N et son contenu ».
    - Request frame : « j'aimerais bien que quelqu'un m'envoie le message N »
      - probablement suivi, quelque temps après, de la data frame contenant la réponse.
    - Error frame : « il y a eu une erreur ».
    - Overload frame : « je n'arrive plus à suivre le rythme ».

CAN

- **Messages**
  - Broadcast, content-adressed
    - Les messages sont à destination de tout le monde.
    - Chaque message possède un identificateur en plus de son contenu. Exemple :
      - message ID 10 : vitesse d'une roue
      - message ID 12 : ordre de freinage d'urgence
    - Les cibles ne traitent que les messages dont l'ID les intéresse (filtres).
    - On parle d'adressage « content-based ».

source : Microchip

- **Trames de données**

  - ID : 11 ou 29 bits
  - control : taille des données utiles
  - data : 0 à 8 octets
  - CRC : 15 + 1 bits
  - SOF / EOF : début / fin de la trame
  - IFS : espace inter-trames
  - bits de stuffing (1 / 5)

- **Trames de données**
  - efficacité du bus

Description	Bits
SOF	1
ID + RTR / SRR +IDE	12 / 32
control	6
data	0...8
CRC	16
ACK	2
EOF	7
IFS	3

Format	Données	Trame	Efficacité
11 bits	1 octet	57 bits	14 %
11 bits	8 octets	114 bits	56 %
29 bits	1 octet	78 bits	10 %
29 bits	8 octet	135 bits	47 %

CAN

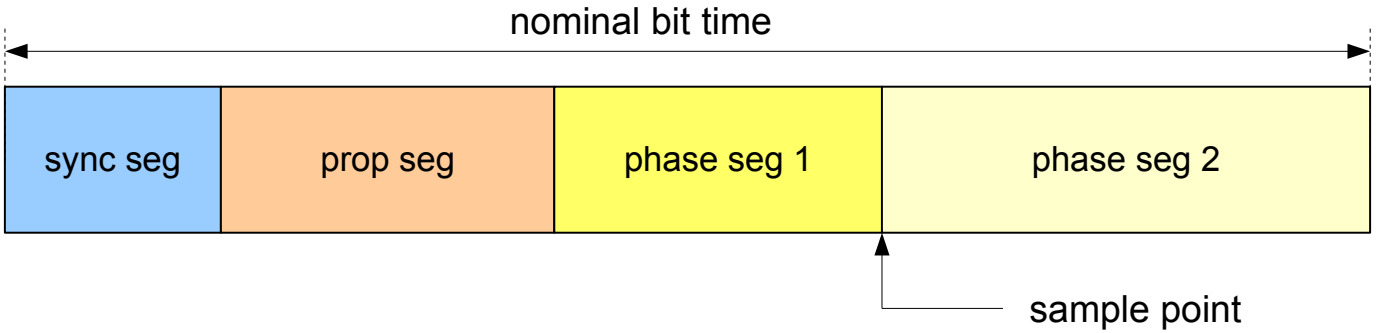
- **Arbitrage**
  - Quand le bus est libre, n'importe quel maître peut commencer à émettre.
  - Si deux maîtres se mettent à émettre en même temps, celui qui envoie le message de plus petite ID gagne (quizz : pourquoi ?).
  - Si deux maître émettent des messages de même ID mais de contenu différents, une erreur sera détectée puis reportée.

source : Microchip

CAN

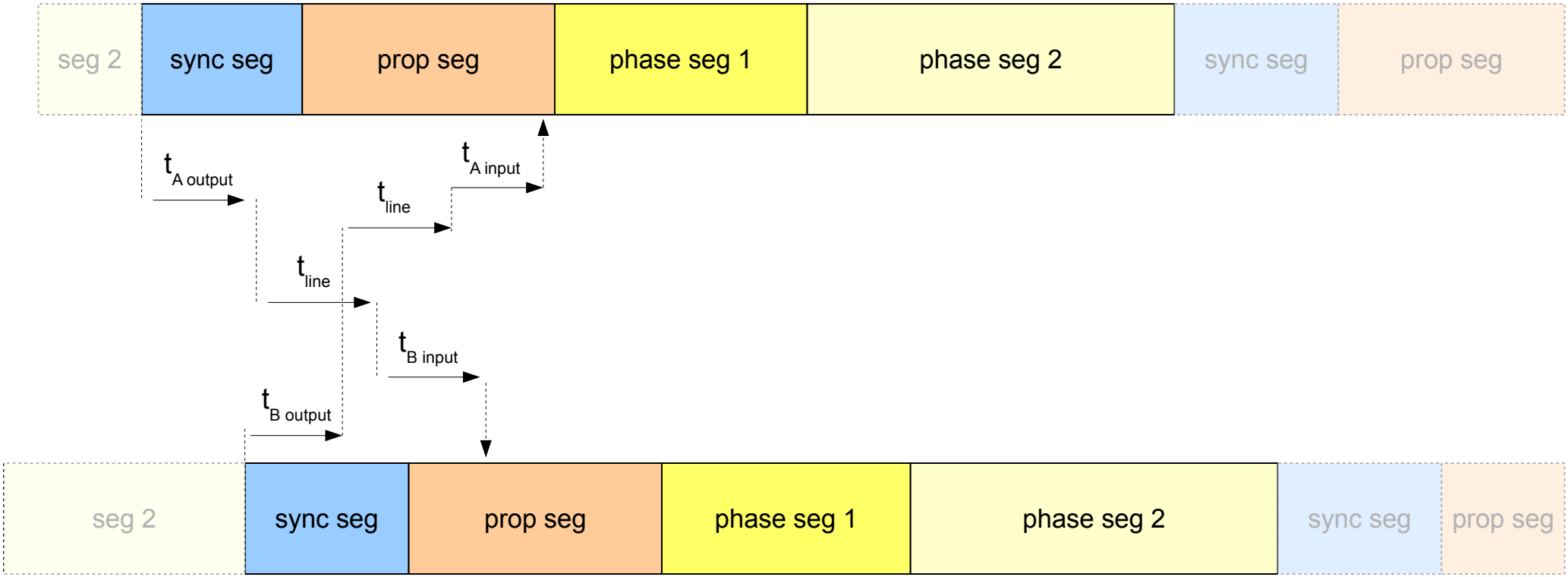
- **Gestion des erreurs**
  - Parti-pris : tous les messages sont corrects.
  - Les nœuds confirment la conformité de chaque message (CRC) par l'envoi d'un ACK (dominant)
  - 3 possibilités :
    - ACK sans trame d'erreur : tout va bien.
    - Pas d'ACK : l'émetteur est fautif.
    - ACK et trame d'erreur : l'erreur est du côté des récepteurs pas contents.
  - Chaque nœud maintient 2 compteurs d'erreurs : RX et TX
    - une émission / réception correcte : décrémente le compteur (lentement),
    - une émission / réception incorrecte : incrémente le compteur (rapidement),
    - selon la valeur des compteurs, le nœud est actif, passif (réception seule) ou déconnecté du bus.
    - ⇒ Détection et confinement des erreurs.

- **Resynchronisation**
  - Bus asynchrone (plésiochrone) : dérive des horloges.
  - Une resynchronisation au niveau du bit :
    - à chaque SOF,
    - à chaque front descendant.
  - Bit divisé en 4 parties, chaque partie en quantum.



  - En fonction de la position des fronts descendants :
    - PHASESEG1 peut être augmenté (de SJW)
    - PHASESEG2 peut être diminué (de SJW)

- **Resynchronisation**



$$PROPSEG \geqslant t_{\text{node\_A}} + 2 \cdot t_{\text{line}} + t_{\text{node\_B}}$$

- **Dérive en fréquence**
  - Soit df la variation maximum de fréquence acceptable :

$$(1-df)\cdot f_{nom} \leqslant f_{osc} \leqslant (1+df)\cdot f_{nom}$$

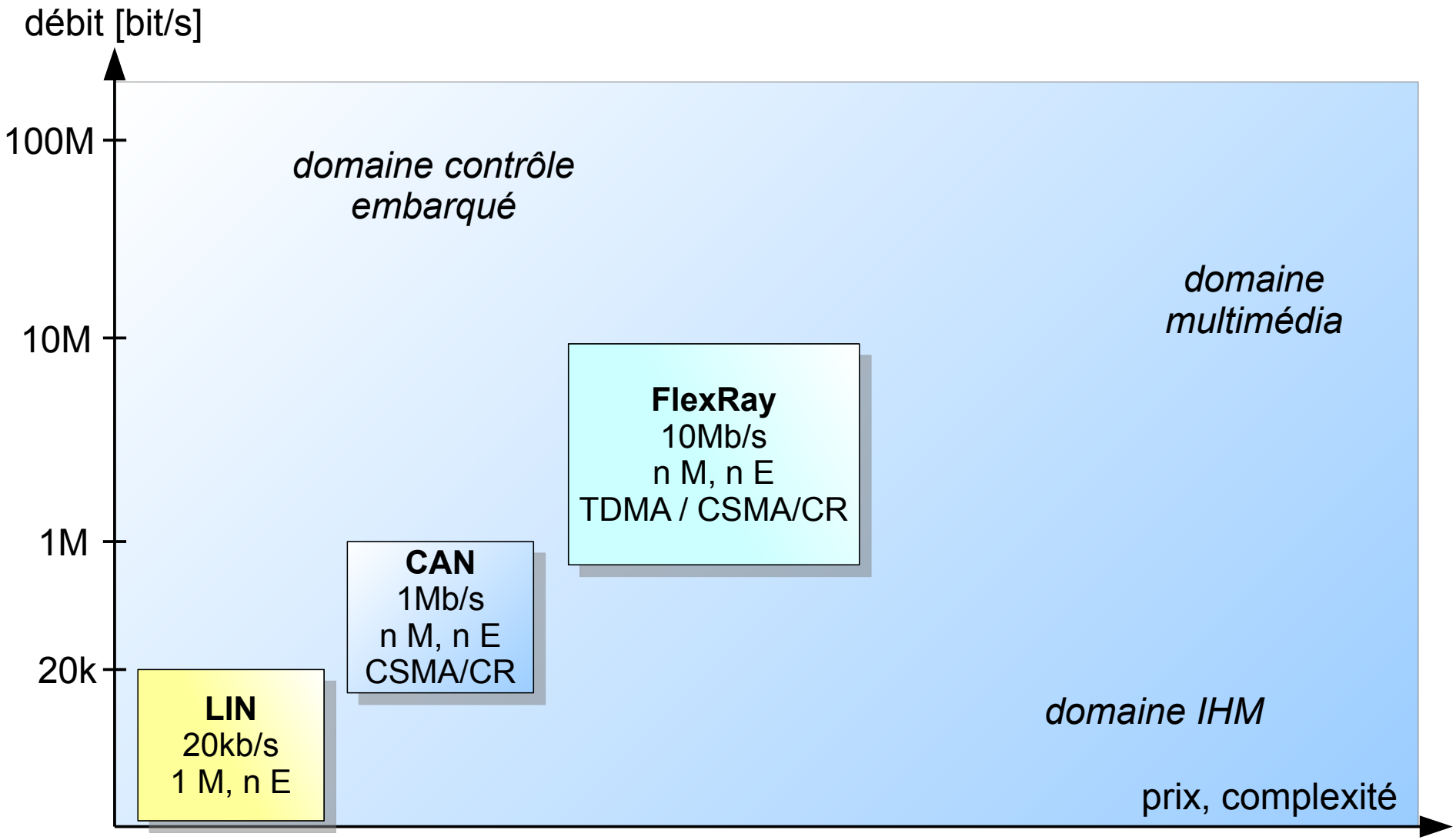
  - Durée maximum sans resynchronisation : 12 bits (flag d'erreur)

$$df \leq \frac{min(PHASESEG1\ ,\ PHASESEG2)}{2\cdot(13\cdot t_{bit} - PHASESEG2)}$$

  - Une erreur de phase doit être totalement éliminée en 10 bits :

$$df \leq \frac{SJW}{2\cdot 10\cdot t_{bit}}$$

Bus automobiles



source : Freescale

FlexRay

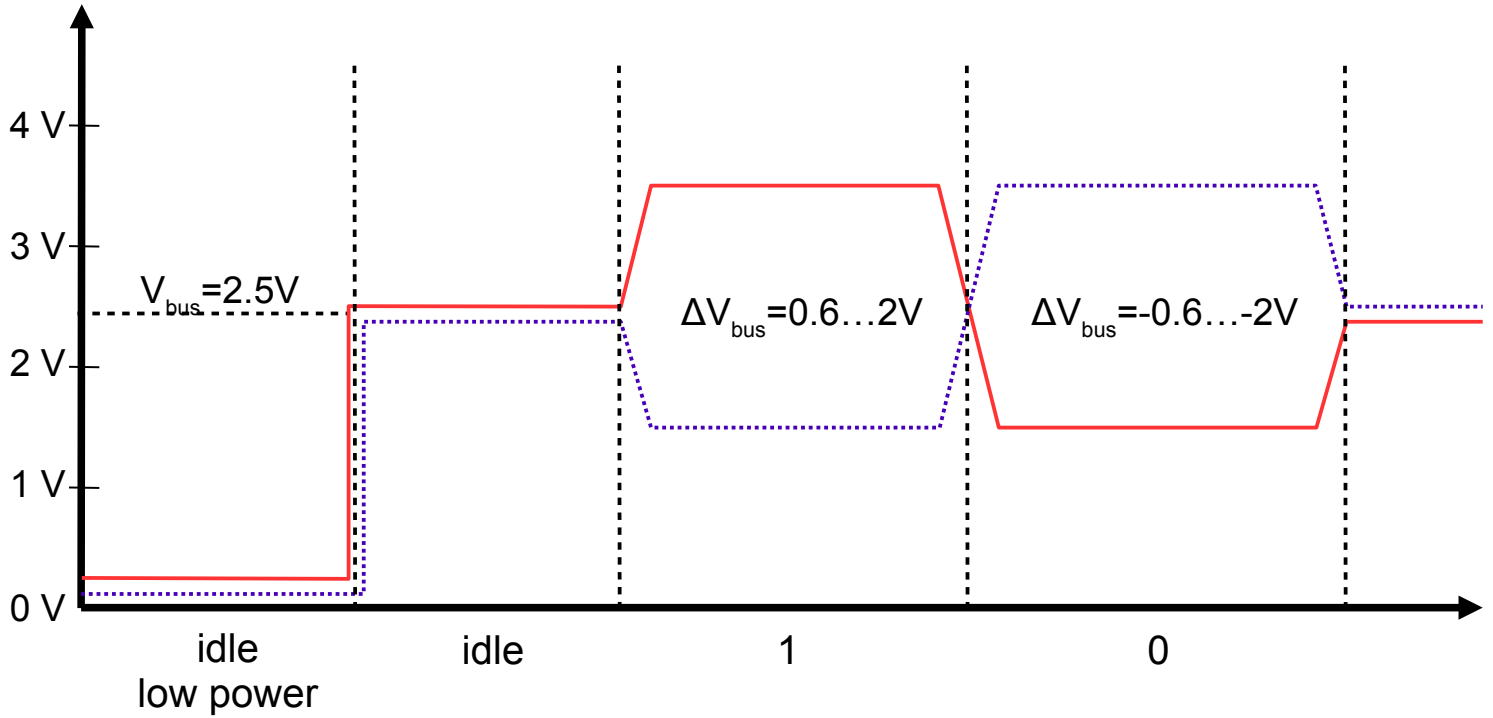
- **FlexRay**
  - Bus développé par le FlexRay consortium : BMW, Bosch, Daimler, Open, Freescale, NXP, Volkswagen…
  - Normalisé en 1999, 2004 (2.0) puis en 2010 (3.0.1).
  - Standard ISO 17458.
  - Non commercial : pas de droits, pas de royalties.
  - Spécifications libres.

FlexRay

- **FlexRay**
  - Temps réel (déterministe, à granularité aussi faible qu'on veut).
  - Bande passante :
    - 10MB/s par canal (4 à 6MB/s utiles)
    - 2 canaux
    - 2 fils par canal
  - Transmission asynchrone.
  - Tolérance aux fautes :
    - Redondance et détection des fautes.
    - X-by-wire.
  - Arbitrage distribué.
  - Souple et extensible.

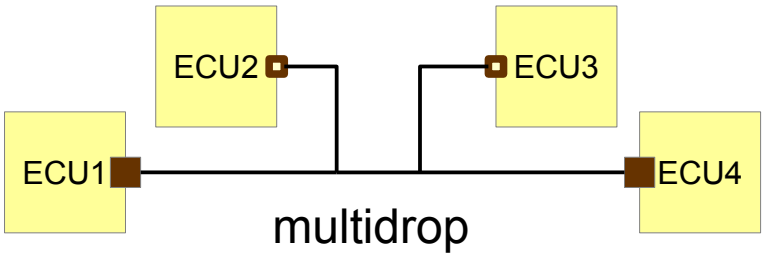
FlexRay

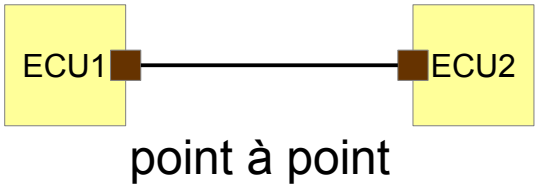
- **Électriquement**
  - chaque canal : paire torsadée, 100Ω
  - DC bus load : 40 à 55Ω

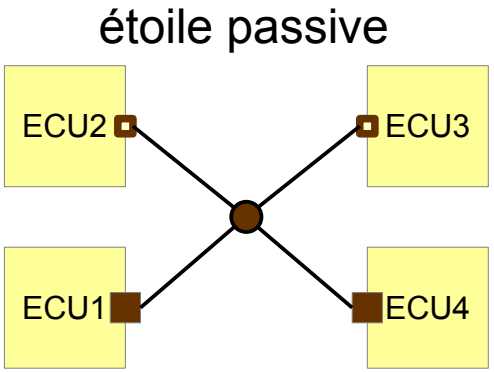


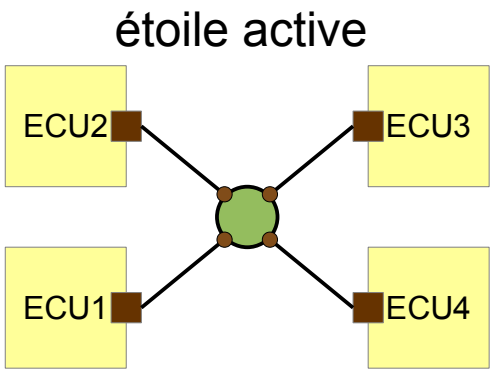
FlexRay

- **Topologies**





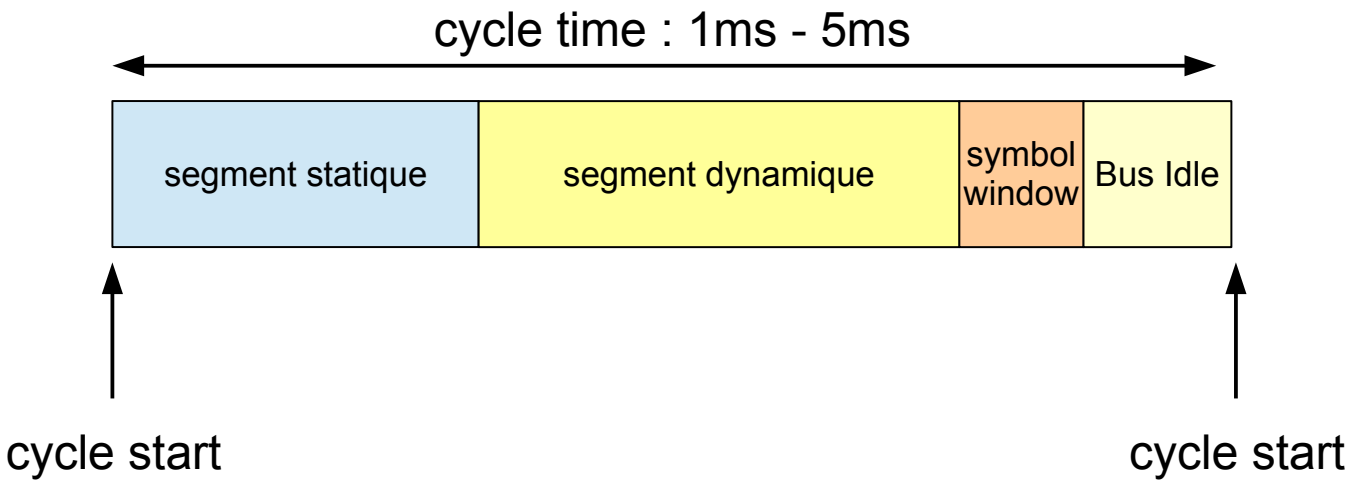




- Plus combinaisons (2 étoiles actives max)
- Étoile active
  - réduit les émissions / réceptions de parasites,
  - confine les fautes.
- 2 canaux, même topologie.

FlexRay

- **Protocole**
  - le temps est découpé en cycles (1 à 5ms typique)
  - chaque cycle comprend
    - segment statique : TDMA, données répétitives,
    - segment dynamique : FTDMA, données épisodiques brèves,
    - symbol window : maintenance / démarrage,
    - network idle time (NIT) : synchronisation.

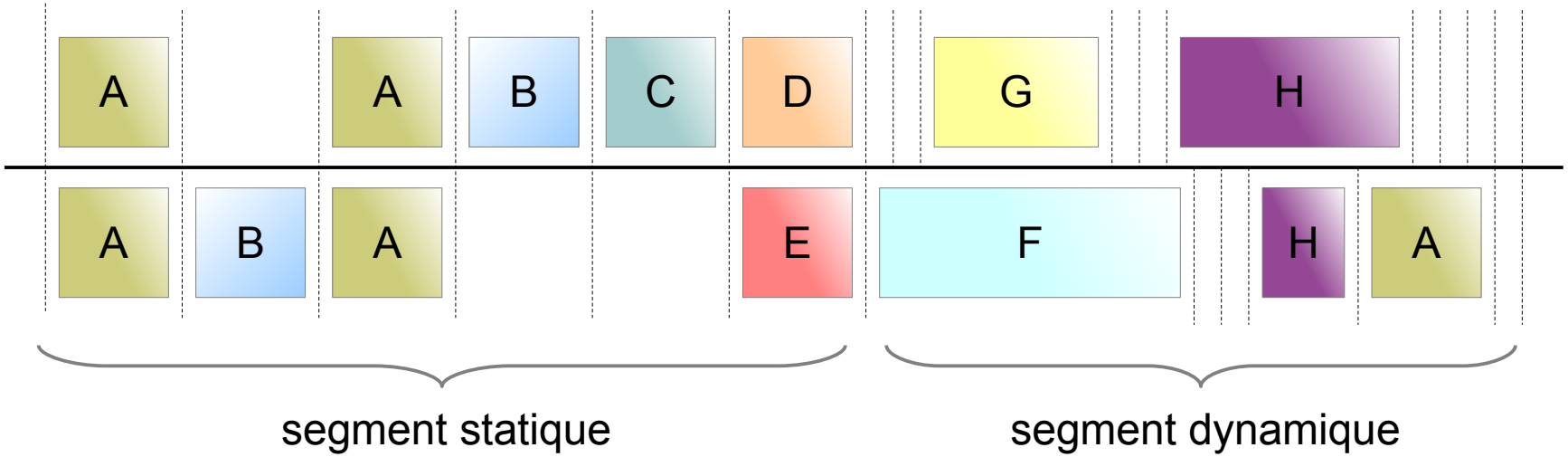


FlexRay

- **Protocole**
  - Les durées des segments statiques et dynamiques sont libres
    - souplesse vis à vis de l'application.
  - Unité de temps : macrotick
    - généralement 1µs,
    - synchronisation de tous les nœuds de façon à ce que les macroticks soient synchrones dans le réseau.

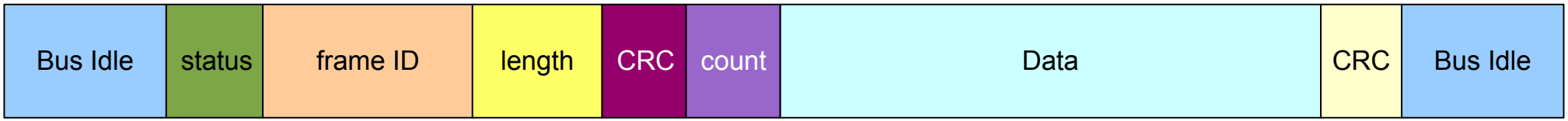
FlexRay

- Segment statique : TDMA
  - Allocation fixe prévue lors de la conception du réseau.
  - Permet des boucles de contrôle à temps de réaction fixe.
- Segment dynamique : CSMA/CA (FTDMA)
  - Slot inutilisé : *minislot*, 1macrotick typ.
  - Slot utilisé : *slot dynamique*, aussi long que nécessaire tant qu'il y a encore assez de place dans le segment.



FlexRay

- **Trames**



- status : 5 bits
- frame ID : 11 bits
- length : 7 bits
- header CRC : 11 bits
- cycle count : 6 bits
- data : 0-254 octets
- CRC : 24 bits

FlexRay

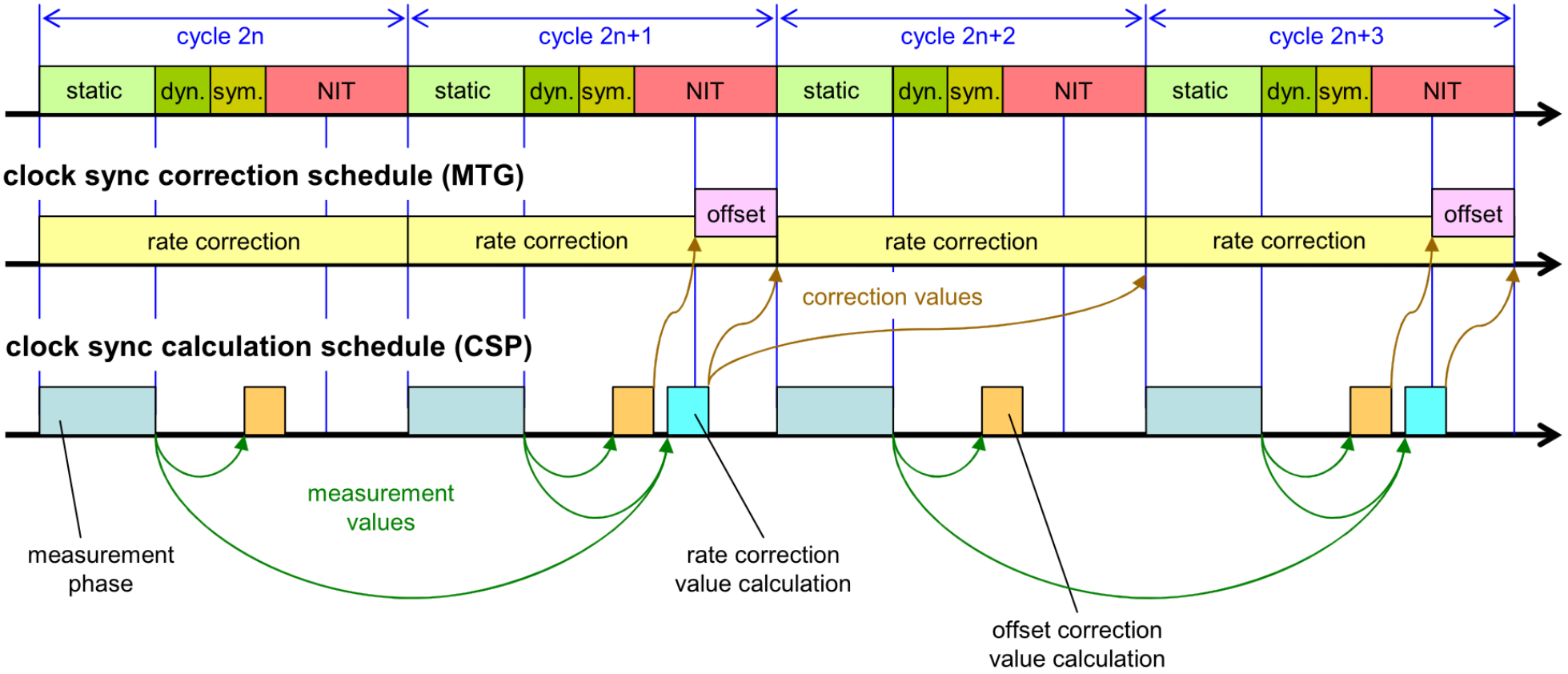
- **Synchronisation**
  - À intervalles réguliers, certains nœuds décident de lancer une phase de synchronisation.
  - un ensemble de nœuds envoient des trames de synchronisation dans un de leurs slots du segment statique
    - Les cycles sont partagés en deux : impairs et pair.
    - Une phase de synchronisation dure deux cycles.
  - Les autres nœuds calculent, pour chaque émetteur :
    - son offset,
    - la durée de son cycle.

FlexRay

- **Synchronisation**
  - Pour offset et débit :
    - En fonction du nombre de nœuds envoyant des trames synchronisation, on enlève un certain nombre des valeurs extrêmes :
      - 1-2 nœuds : 0 valeur
      - 3-7 nœuds : 1 valeur
      - 8-15 nœuds : 2 valeurs
    - On calcule la moyenne des extrêmes restants.
  - La phase est corrigée avec l'offset moyen lors du NIT.
  - La durée du cycle est corrigée avec la durée moyenne.

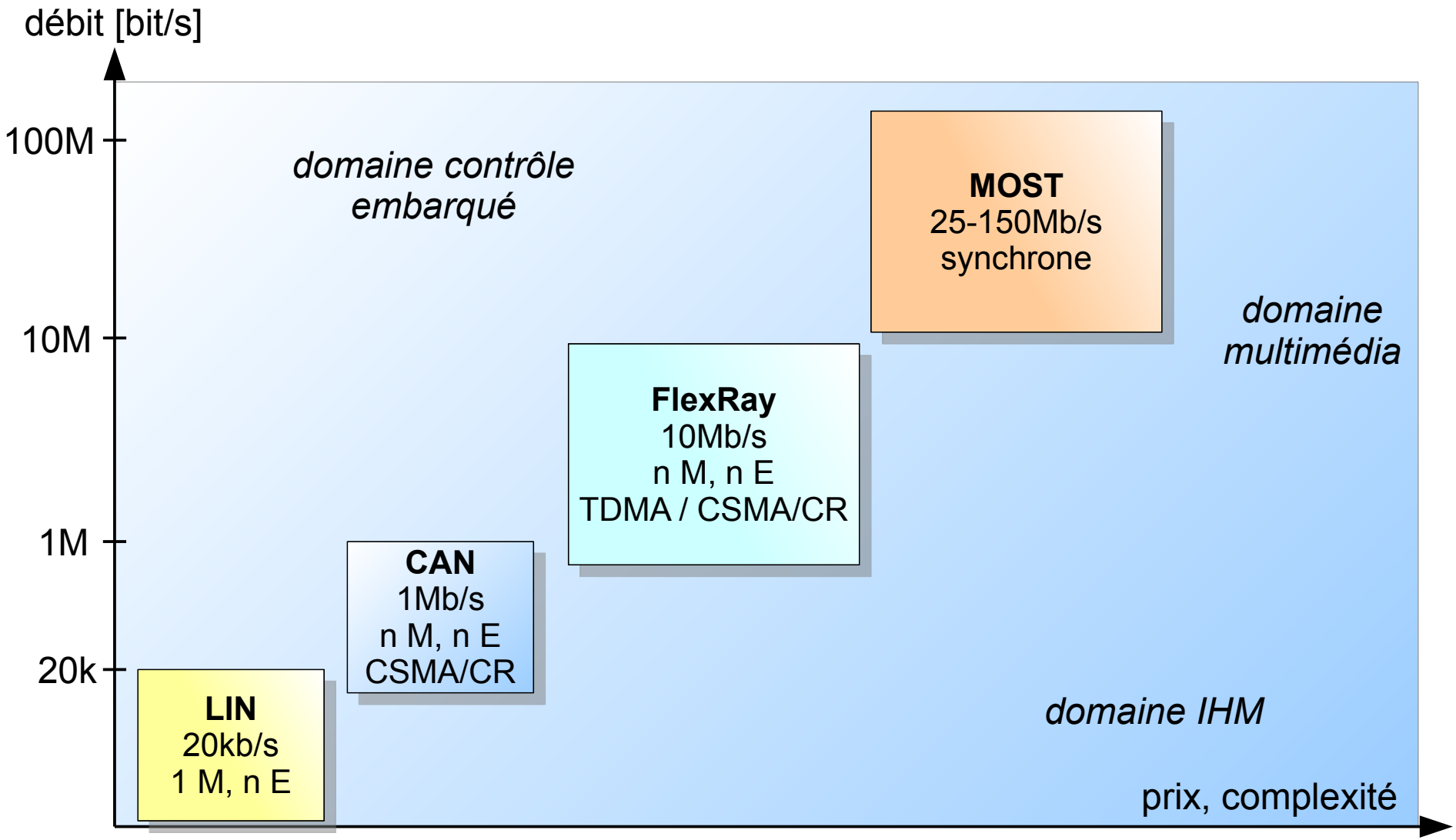
FlexRay

- **Synchronisation**



source : FlexRay specification 3.0.1

Bus automobiles



source : Freescale

Licence de droits d'usage

Contexte académique } **sans modification**

Par le téléchargement ou la consultation de ce document, l'utilisateur accepte la licence d'utilisation qui y est attachée, telle que détaillée dans les dispositions suivantes, et s'engage à la respecter intégralement.

La licence confère à l'utilisateur un droit d'usage sur le document consulté ou téléchargé, totalement ou en partie, dans les conditions définies ci-après, et à l'exclusion de toute utilisation commerciale.

Le droit d'usage défini par la licence autorise un usage dans un cadre académique, par un utilisateur donnant des cours dans un établissement d'enseignement secondaire ou supérieur et à l'exclusion expresse des formations commerciales et notamment de formation continue. Ce droit comprend :

- le droit de reproduire tout ou partie du document sur support informatique ou papier,
- le droit de diffuser tout ou partie du document à destination des élèves ou étudiants.

Aucune modification du document dans son contenu, sa forme ou sa présentation n'est autorisé.

Les mentions relatives à la source du document et/ou à son auteur doivent être conservées dans leur intégralité.

Le droit d'usage défini par la licence est personnel, non exclusif et non transmissible.

Tout autre usage que ceux prévus par la licence est soumis à autorisation préalable et expresse de l'auteur :

alexis.polti@telecom-paristech.fr