Conception en logique asynchrone

Jérémie Hamon
Laboratoire TIMA

Ecole IN2P3 « Electronique Numérique 2010 »
17 Septembre 2010 - Oléron

Laboratoire TIMA

http://tima.imag.fr

- Techniques de l'Informatique et de la Microélectronique pour l'Architecture des systèmes intégrés
- ~ 100 personnes
- 6 Groupes de Recherche :
  - ARIS : Architecture for Robust and complex Integrated Systems
  - CIS : Concurrent Integrated Systems
  - MNS : Micro and Nano Systems
  - RMS : Reliable Mixed-signal Systems
  - SLS : System Level Synthesis
  - VDS : Verification & Modeling of Digital Systems

jeremie.hamon@imag.fr

Groupe CIS

http://tima.imag.fr/cis

- Créé en 1998 par Pr. Marc Renaudin
- Codirigé par Dr. Laurent Fesquet et Dr. Gilles Sicard
- Conception de circuits et systèmes asynchrones
  - Outils et méthodes de conception (TAST)
  - Tima Asynchronous Library (TAL)
  - Basse consommation (Processeurs, Réseaux de capteurs,…)
  - Faible bruit / EMI
  - Sécurité (Crypto-processeurs, FPGA sécurisé,…)
  - Fiabilité / Sureté de fonctionnement (Tolérance aux fautes, preuve formelle,…)
  - Traitement du signal asynchrone (ACAN, échantillonnage irrégulier,…)
- Imageurs CMOS

Première partie

Introduction à la logique asynchrone

- Approche synchrone : Intérêt… et limitations
- Concepts de base de la logique asynchrone
  - Synchronisation locale
  - Protocoles de communication
  - Codage des données
  - Signaux d'acquittement
- Classe des circuits asynchrones
- Propriétés
  - Absence d'horloge
  - Calcul en temps minimal
  - Technologies déca-nanométrique
  - Modularité
  - Faible consommation
  - Faible EMI

jeremie.hamon@imag.fr

Première partie

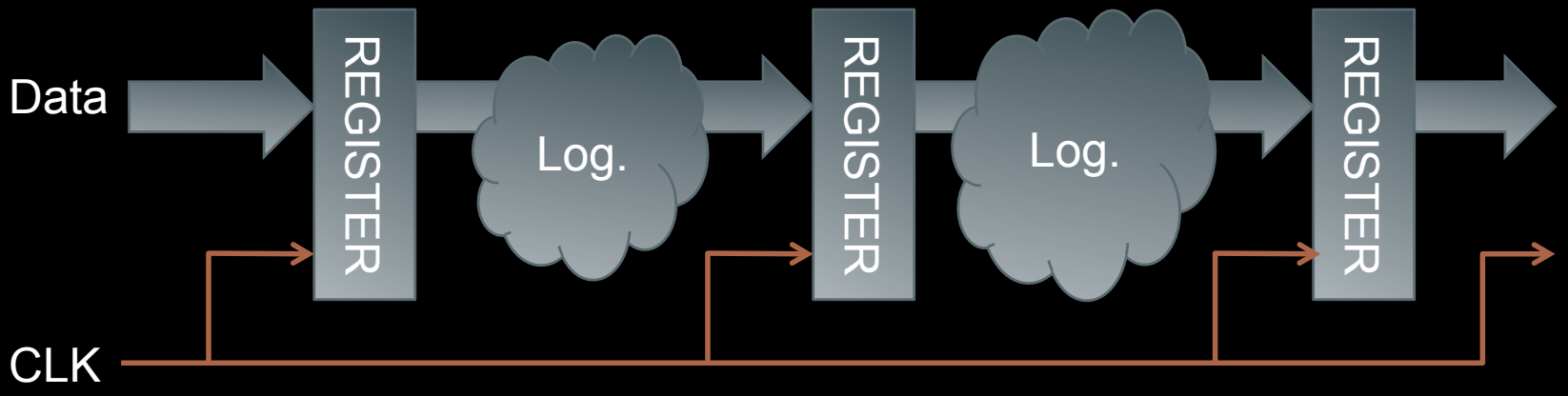
Introduction à la logique asynchrone

- Approche synchrone : Intérêt… et limitations
- Concepts de base de la logique asynchrone
  - Synchronisation locale
  - Protocoles de communication
  - Codage des données
  - Signaux d'acquittement
- Classe des circuits asynchrones
- Propriétés
  - Absence d'horloge
  - Calcul en temps minimal
  - Technologies déca-nanométrique
  - Modularité
  - Faible consommation
  - Faible EMI

jeremie.hamon@imag.fr

Approche synchrone

Intérêt…



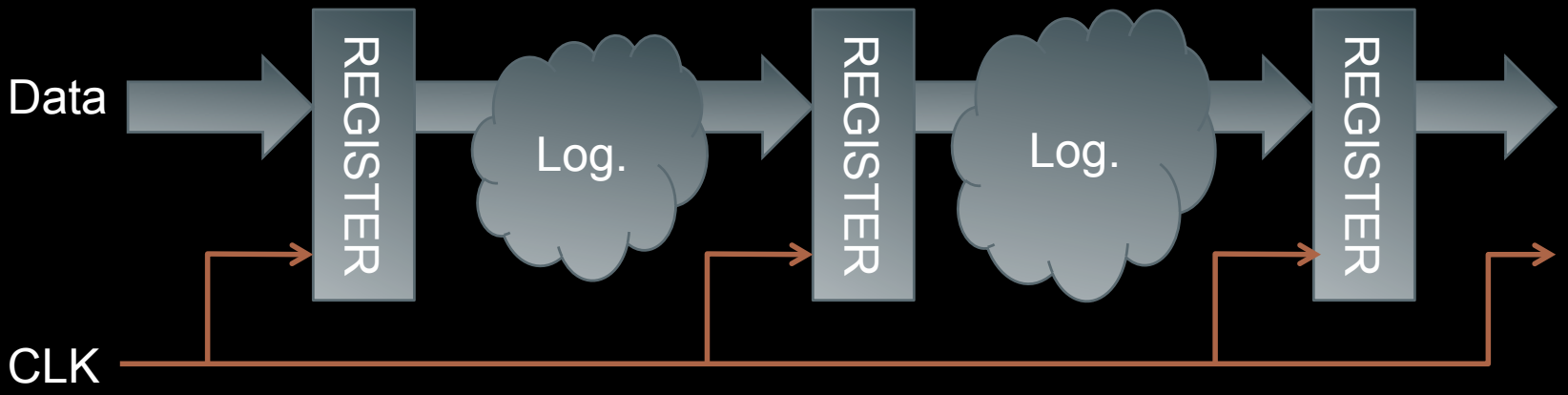
- Horloge globale :
  - Hypothèse de discrétisation temporelle
  - Conception des blocs combinatoires simple (aléas logiques filtrés)

- Contrainte globale : $T_{clk} > Max(delais_{log})$

jeremie.hamon@imag.fr

Approche synchrone

… et limitations



- Distribution de l'horloge
- Performances
- Modularité
- Rayonnement électromagnétique
- Sensibilité aux variations PVT
- …

jeremie.hamon@imag.fr

Première partie

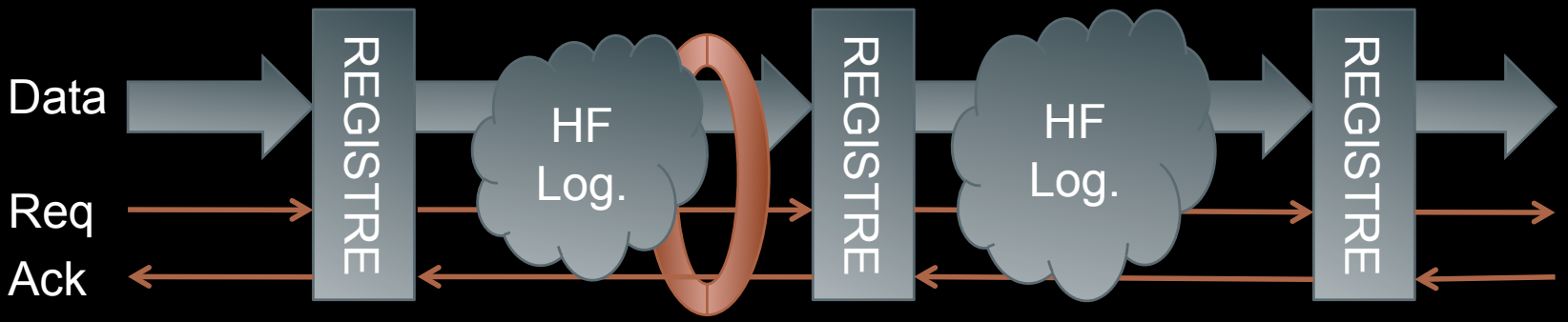
Introduction à la logique asynchrone

- Approche synchrone : Intérêt… et limitations
- Concepts de base de la logique asynchrone
  - Synchronisation locale
  - Protocoles de communication
  - Codage des données
  - Signaux d'acquittement
- Classe des circuits asynchrones
- Propriétés
  - Absence d'horloge
  - Calcul en temps minimal
  - Technologies déca-nanométrique
  - Modularité
  - Faible consommation
  - Faible EMI

jeremie.hamon@imag.fr

Concepts de base

Synchronisation locale

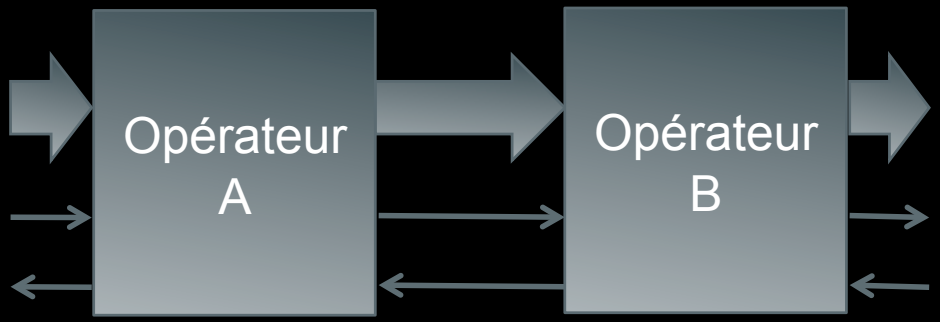


- Opérateurs reliés entre eux par des canaux de communication
- Signalisation bi-directionnelle (requête/acquittement)
- Fonctionnement de type « flot de données »
- Contrainte : Logique sans aléas

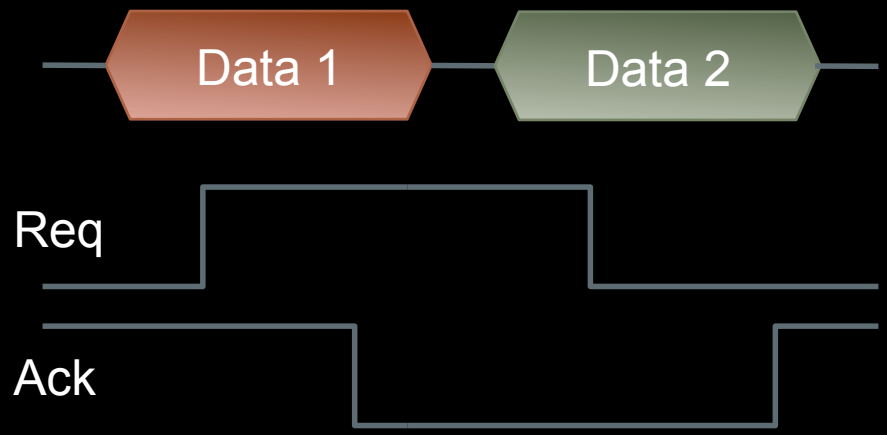
jeremie.hamon@imag.fr

Concepts de base

Protocoles de communication : 2 phases (NRZ)



- 1ère Phase (Récepteur)
  - Détection de la requête
  - Consommation de la donnée
  - Acquittement

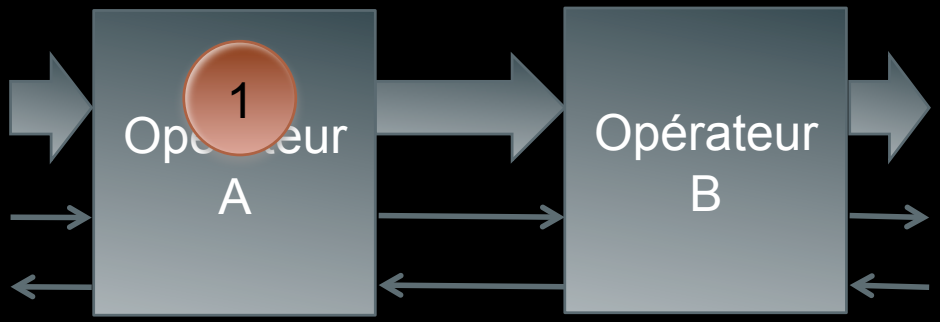


- 2ème Phase (Emetteur)
  - Détection de l'acquittement
  - Production d'une nouvelle donnée
  - Requête

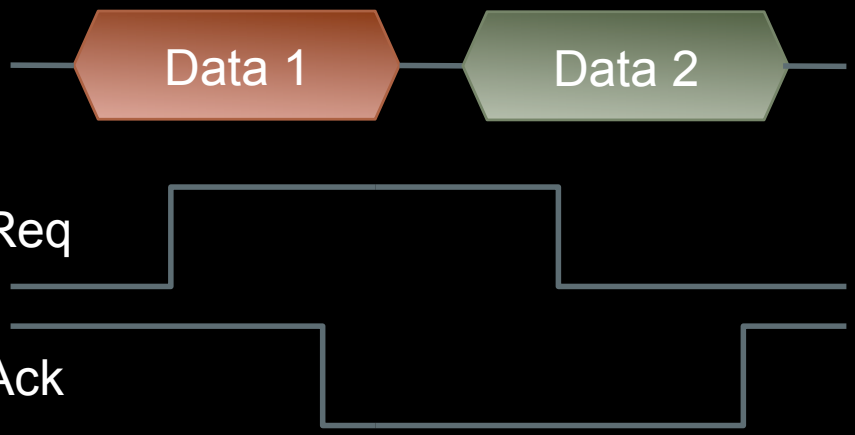
jeremie.hamon@imag.fr

Concepts de base

Protocoles de communication : 2 phases (NRZ)



- 1ère Phase (Récepteur)
  - Détection de la requête
  - Consommation de la donnée
  - Acquittement

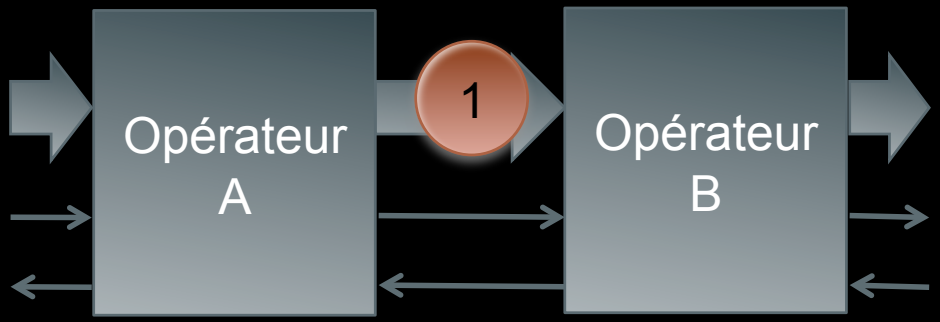


- 2ème Phase (Emetteur)
  - Détection de l'acquittement
  - Production d'une nouvelle donnée
  - Requête

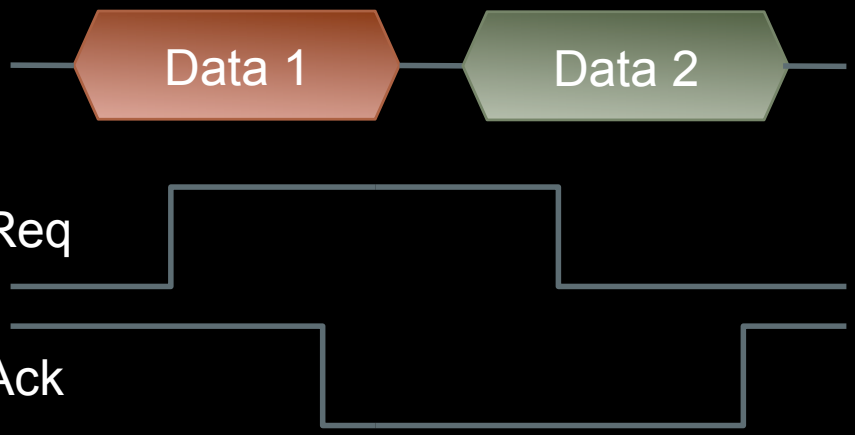
jeremie.hamon@imag.fr

Concepts de base

Protocoles de communication : 2 phases (NRZ)



- 1ère Phase (Récepteur)
  - Détection de la requête
  - Consommation de la donnée
  - Acquittement

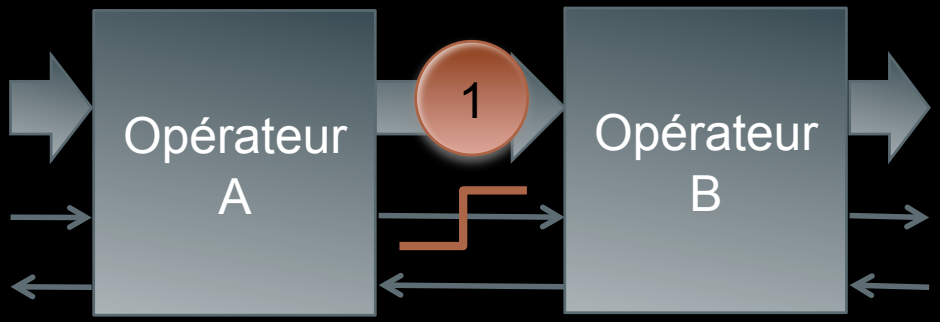


- 2ème Phase (Emetteur)
  - Détection de l'acquittement
  - Production d'une nouvelle donnée
  - Requête

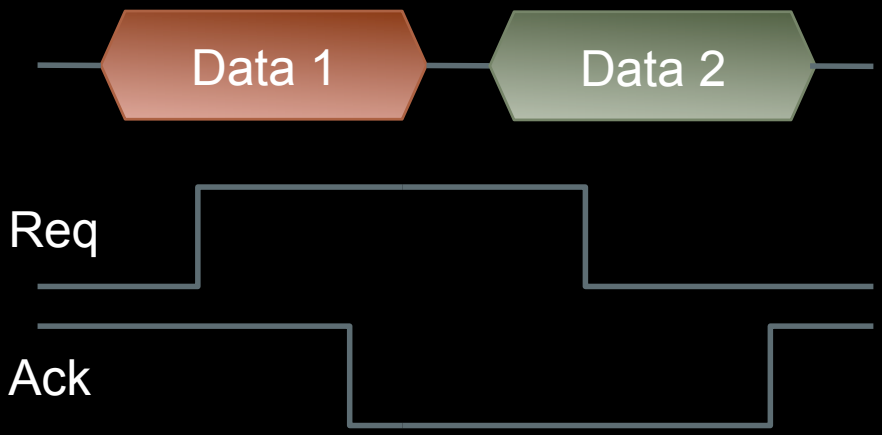
jeremie.hamon@imag.fr

Concepts de base

Protocoles de communication : 2 phases (NRZ)



- 1ère Phase (Récepteur)
  - Détection de la requête
  - Consommation de la donnée
  - Acquittement

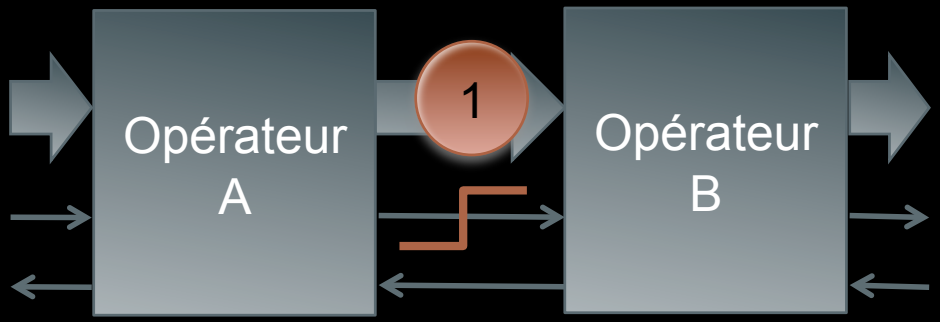


- 2ème Phase (Emetteur)
  - Détection de l'acquittement
  - Production d'une nouvelle donnée
  - Requête

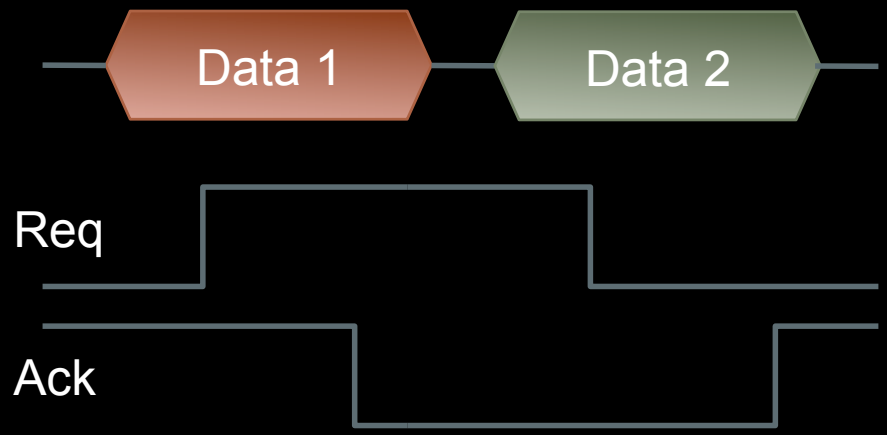
jeremie.hamon@imag.fr

Concepts de base

Protocoles de communication : 2 phases (NRZ)



- 1ère Phase (Récepteur)
  - Détection de la requête
  - Consommation de la donnée
  - Acquittement

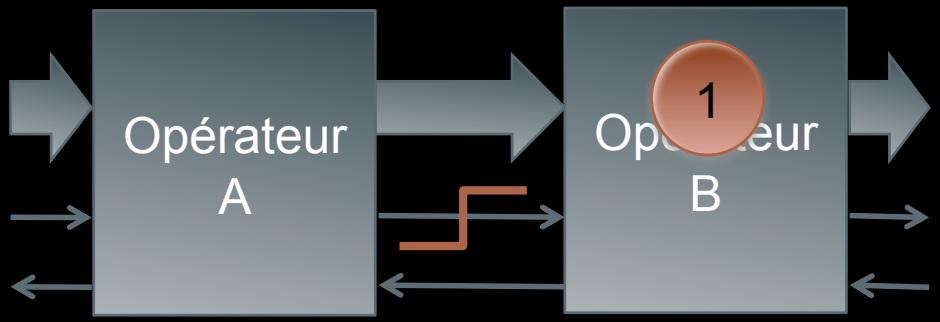


- 2ème Phase (Emetteur)
  - Détection de l'acquittement
  - Production d'une nouvelle donnée
  - Requête

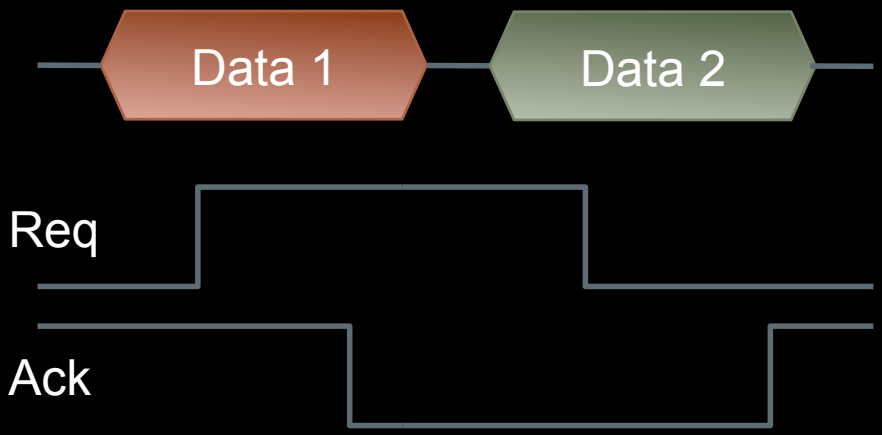
jeremie.hamon@imag.fr

Concepts de base

Protocoles de communication : 2 phases (NRZ)



- 1ère Phase (Récepteur)
  - Détection de la requête
  - Consommation de la donnée
  - Acquittement

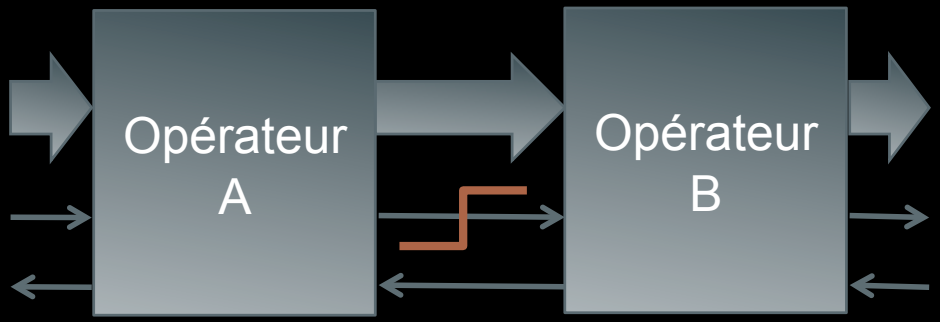


- 2ème Phase (Emetteur)
  - Détection de l'acquittement
  - Production d'une nouvelle donnée
  - Requête

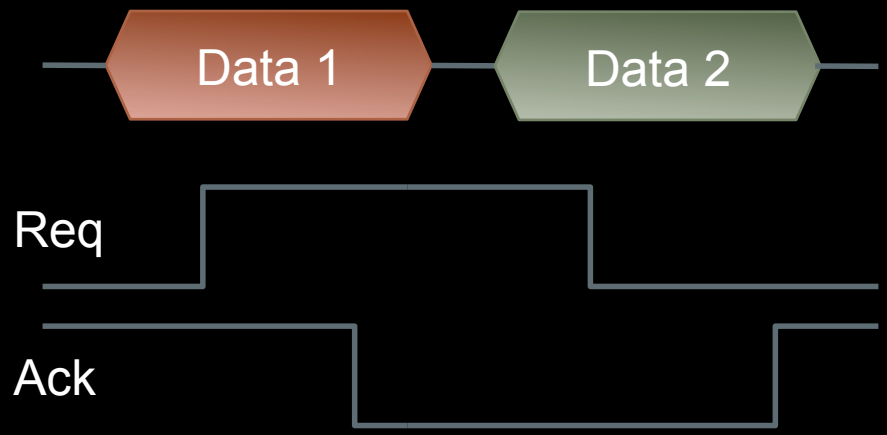
jeremie.hamon@imag.fr

Concepts de base

Protocoles de communication : 2 phases (NRZ)



- 1ère Phase (Récepteur)
  - Détection de la requête
  - Consommation de la donnée
  - Acquittement

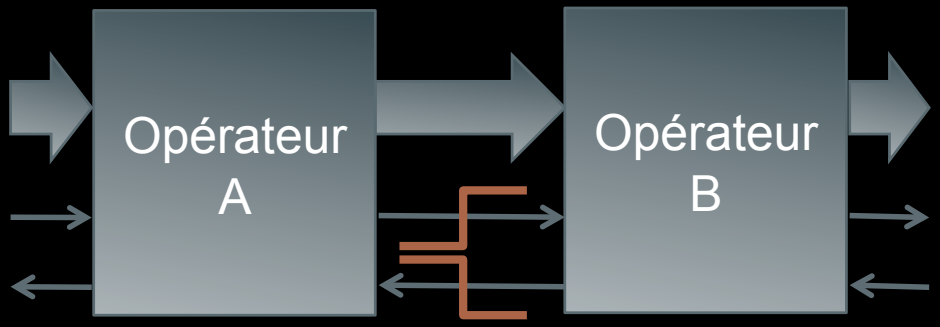


- 2ème Phase (Emetteur)
  - Détection de l'acquittement
  - Production d'une nouvelle donnée
  - Requête

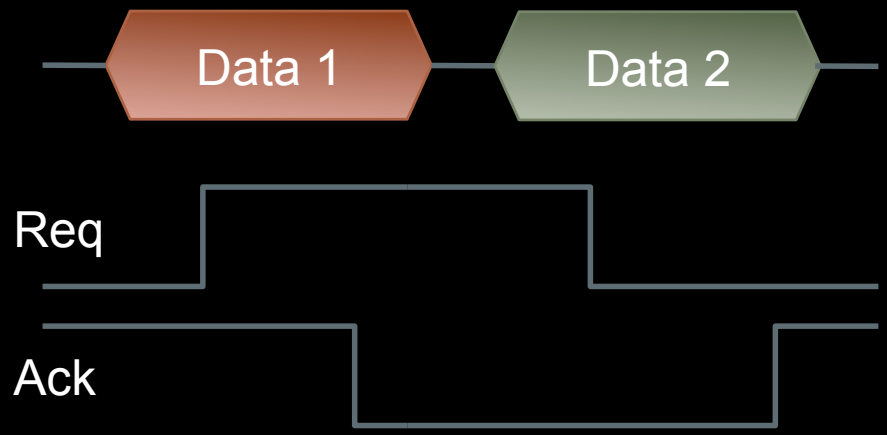
jeremie.hamon@imag.fr

Concepts de base

Protocoles de communication : 2 phases (NRZ)



- 1ère Phase (Récepteur)
  - Détection de la requête
  - Consommation de la donnée
  - Acquittement

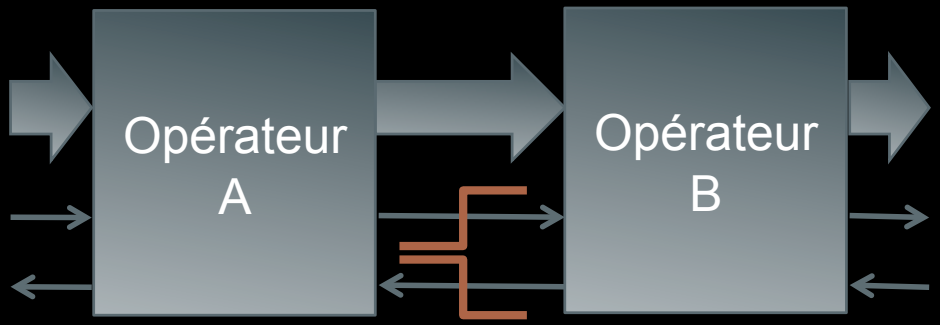


- 2ème Phase (Emetteur)
  - Détection de l'acquittement
  - Production d'une nouvelle donnée
  - Requête

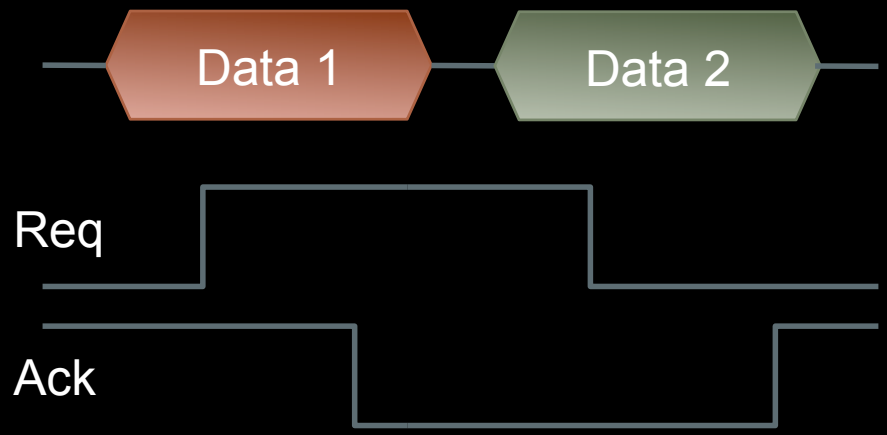
jeremie.hamon@imag.fr

Concepts de base

Protocoles de communication : 2 phases (NRZ)



- 1ère Phase (Récepteur)
  - Détection de la requête
  - Consommation de la donnée
  - Acquittement

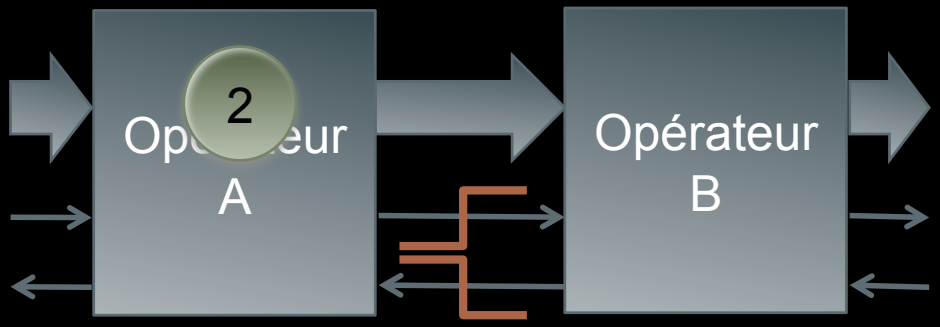


- 2ème Phase (Emetteur)
  - Détection de l'acquittement
  - Production d'une nouvelle donnée
  - Requête

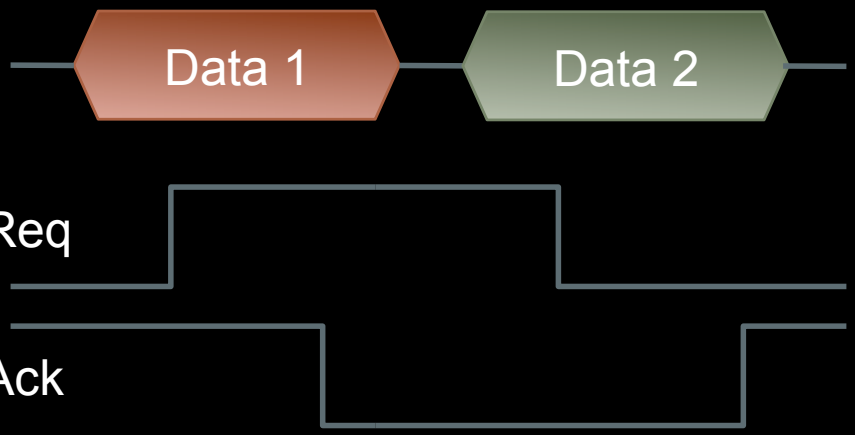
jeremie.hamon@imag.fr

Concepts de base

Protocoles de communication : 2 phases (NRZ)



- 1ère Phase (Récepteur)
  - Détection de la requête
  - Consommation de la donnée
  - Acquittement

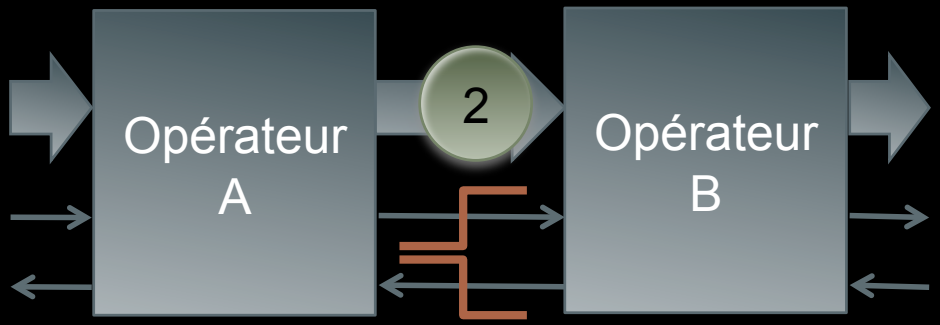


- 2ème Phase (Emetteur)
  - Détection de l'acquittement
  - Production d'une nouvelle donnée
  - Requête

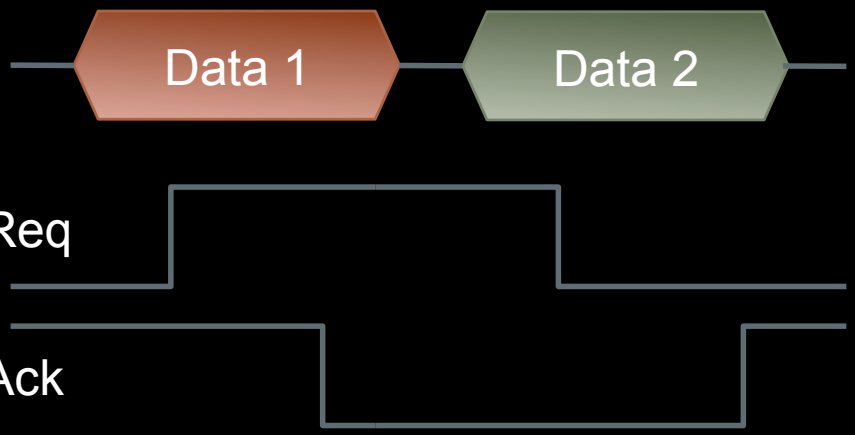
jeremie.hamon@imag.fr

Concepts de base

Protocoles de communication : 2 phases (NRZ)



- 1ère Phase (Récepteur)
  - Détection de la requête
  - Consommation de la donnée
  - Acquittement

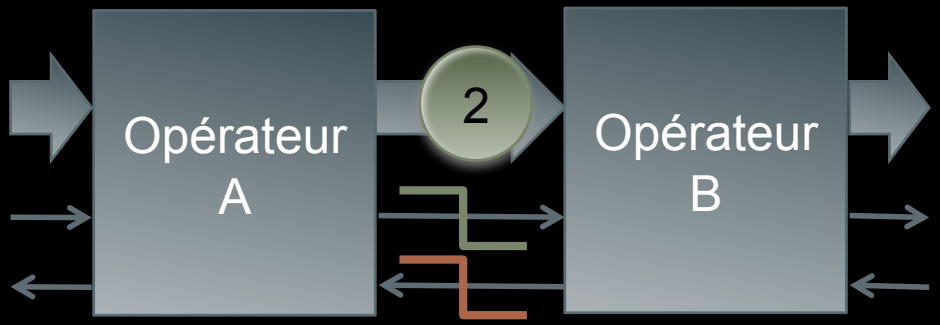


- 2ème Phase (Emetteur)
  - Détection de l'acquittement
  - Production d'une nouvelle donnée
  - Requête

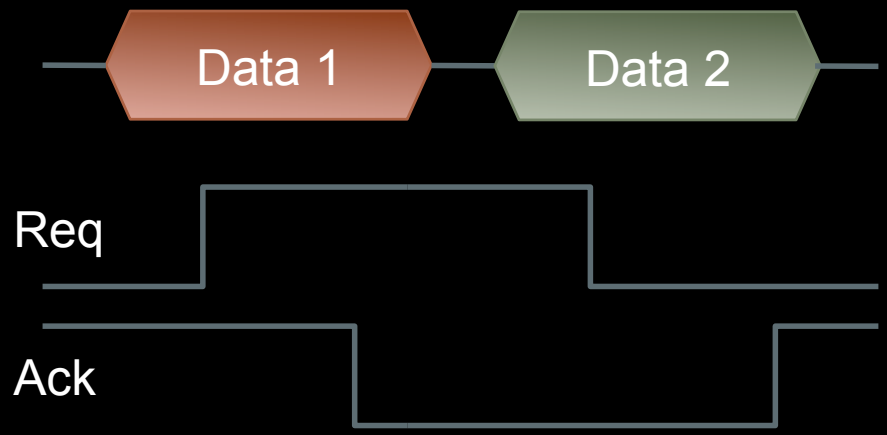
jeremie.hamon@imag.fr

Concepts de base

Protocoles de communication : 2 phases (NRZ)



- 1ère Phase (Récepteur)
  - Détection de la requête
  - Consommation de la donnée
  - Acquittement

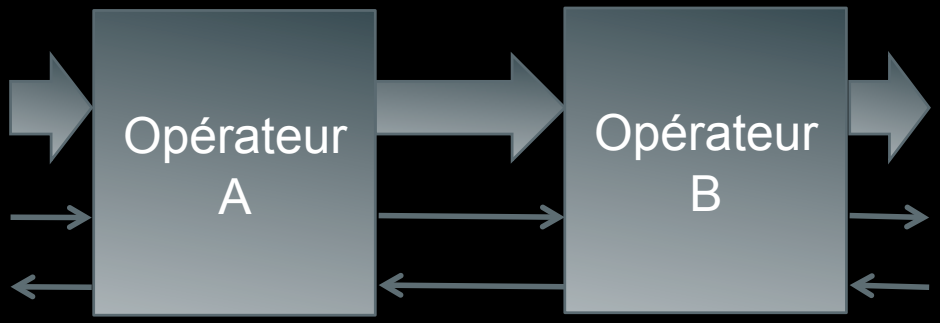


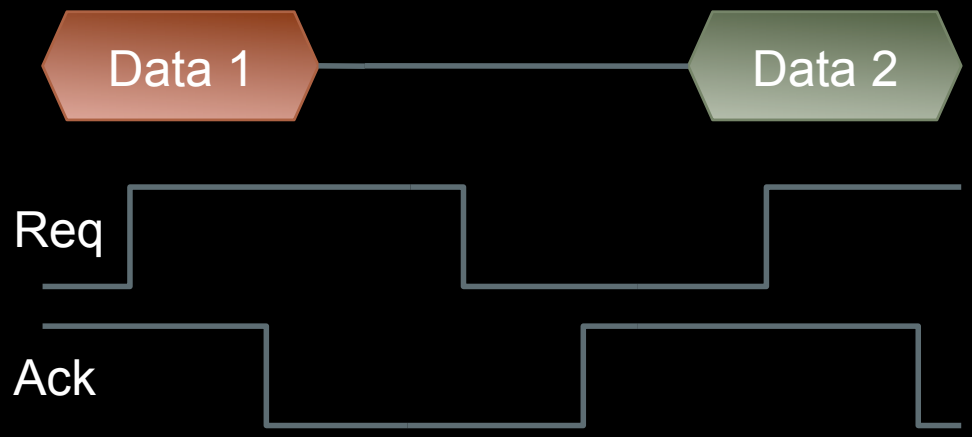
- 2ème Phase (Emetteur)
  - Détection de l'acquittement
  - Production d'une nouvelle donnée
  - Requête

jeremie.hamon@imag.fr

Concepts de base

Protocoles de communication : 4 phases (RZ)



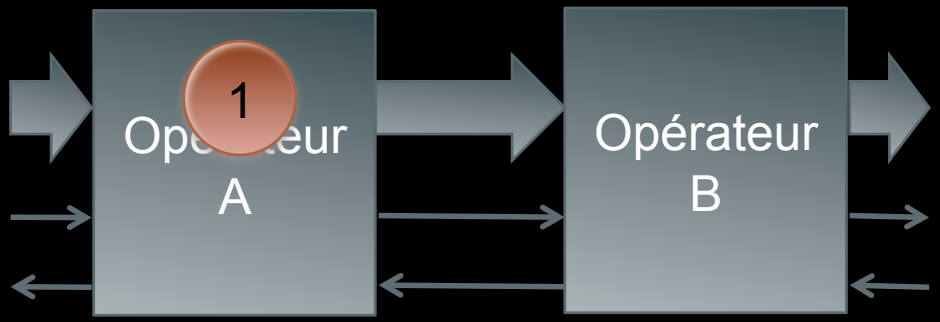


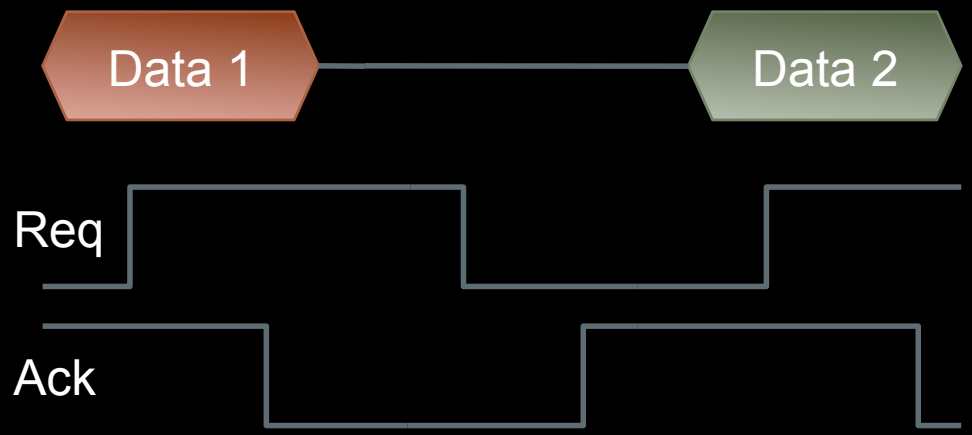
- 1ère Phase (Récepteur)
  - Détection de la requête
  - Consommation de la donnée
  - Acquittement
- 2ème Phase (Emetteur)
  - Détection de l'acquittement
  - Remise à zéro de la requête
- 3ème Phase (Récepteur)
  - Détection de la remise à zéro de la requête
  - Remise à zéro de l'acquittement
- 4ème Phase (Emetteur)
  - Détection de la remise à zéro de l'acquittement
  - Production d'une nouvelle donnée
  - Requête

jeremie.hamon@imag.fr

Concepts de base

Protocoles de communication : 4 phases (RZ)



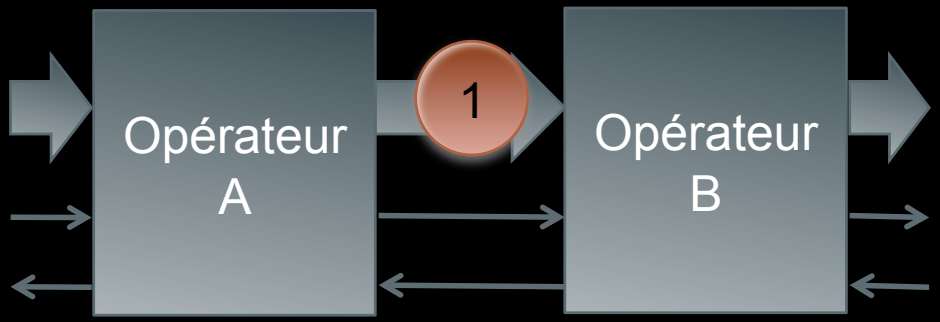


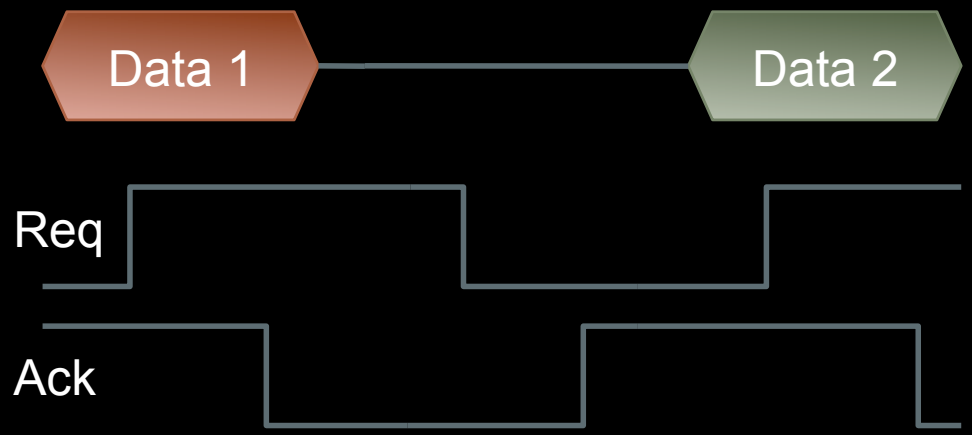
- 1ère Phase (Récepteur)
  - Détection de la requête
  - Consommation de la donnée
  - Acquittement
- 2ème Phase (Emetteur)
  - Détection de l'acquittement
  - Remise à zéro de la requête
- 3ème Phase (Récepteur)
  - Détection de la remise à zéro de la requête
  - Remise à zéro de l'acquittement
- 4ème Phase (Emetteur)
  - Détection de la remise à zéro de l'acquittement
  - Production d'une nouvelle donnée
  - Requête

jeremie.hamon@imag.fr

Concepts de base

Protocoles de communication : 4 phases (RZ)



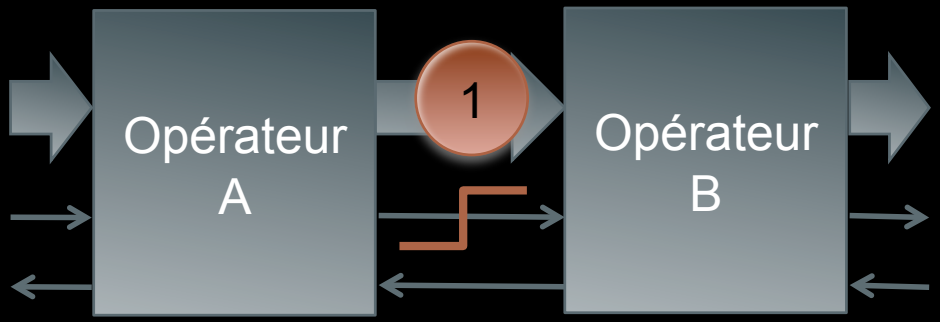


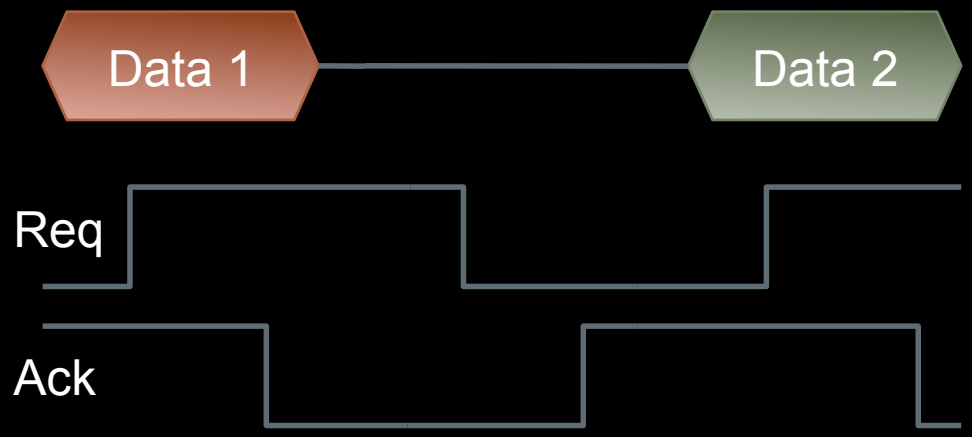
- 1ère Phase (Récepteur)
  - Détection de la requête
  - Consommation de la donnée
  - Acquittement
- 2ème Phase (Emetteur)
  - Détection de l'acquittement
  - Remise à zéro de la requête
- 3ème Phase (Récepteur)
  - Détection de la remise à zéro de la requête
  - Remise à zéro de l'acquittement
- 4ème Phase (Emetteur)
  - Détection de la remise à zéro de l'acquittement
  - Production d'une nouvelle donnée
  - Requête

jeremie.hamon@imag.fr

Concepts de base

Protocoles de communication : 4 phases (RZ)



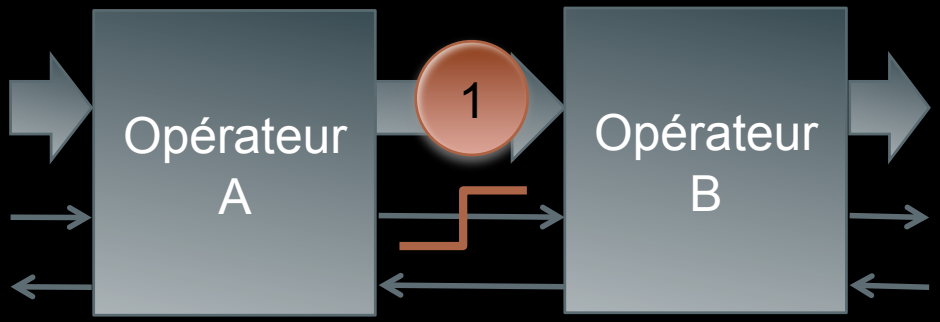


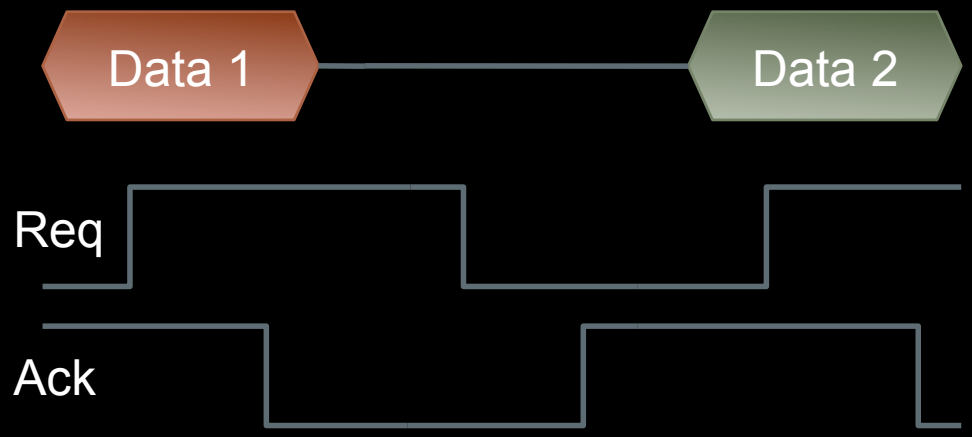
- 1ère Phase (Récepteur)
  - Détection de la requête
  - Consommation de la donnée
  - Acquittement
- 2ème Phase (Emetteur)
  - Détection de l'acquittement
  - Remise à zéro de la requête
- 3ème Phase (Récepteur)
  - Détection de la remise à zéro de la requête
  - Remise à zéro de l'acquittement
- 4ème Phase (Emetteur)
  - Détection de la remise à zéro de l'acquittement
  - Production d'une nouvelle donnée
  - Requête

jeremie.hamon@imag.fr

Concepts de base

Protocoles de communication : 4 phases (RZ)



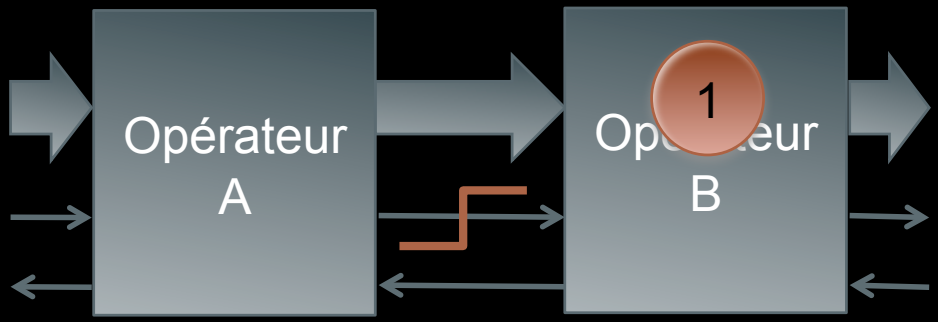


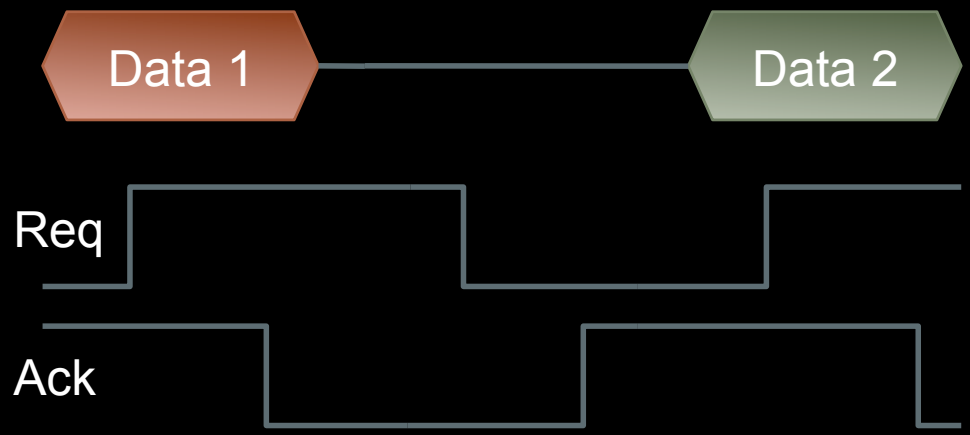
- 1ère Phase (Récepteur)
  - Détection de la requête
  - Consommation de la donnée
  - Acquittement
- 2ème Phase (Emetteur)
  - Détection de l'acquittement
  - Remise à zéro de la requête
- 3ème Phase (Récepteur)
  - Détection de la remise à zéro de la requête
  - Remise à zéro de l'acquittement
- 4ème Phase (Emetteur)
  - Détection de la remise à zéro de l'acquittement
  - Production d'une nouvelle donnée
  - Requête

jeremie.hamon@imag.fr

Concepts de base

Protocoles de communication : 4 phases (RZ)



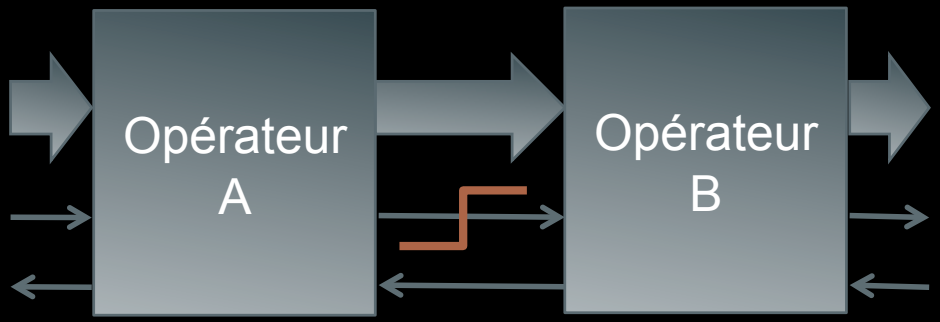


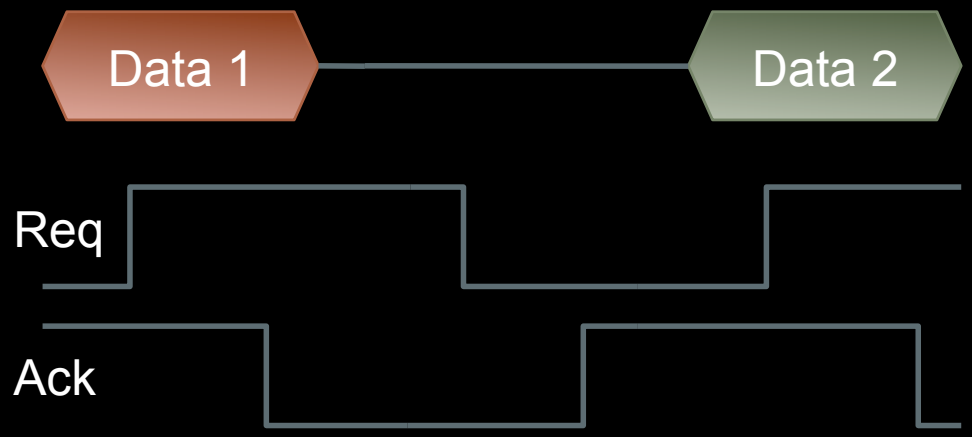
- 1ère Phase (Récepteur)
  - Détection de la requête
  - Consommation de la donnée
  - Acquittement
- 2ème Phase (Emetteur)
  - Détection de l'acquittement
  - Remise à zéro de la requête
- 3ème Phase (Récepteur)
  - Détection de la remise à zéro de la requête
  - Remise à zéro de l'acquittement
- 4ème Phase (Emetteur)
  - Détection de la remise à zéro de l'acquittement
  - Production d'une nouvelle donnée
  - Requête

jeremie.hamon@imag.fr

Concepts de base

Protocoles de communication : 4 phases (RZ)



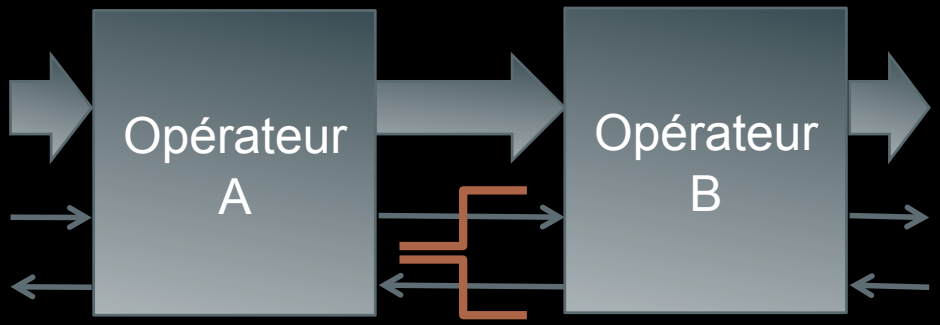


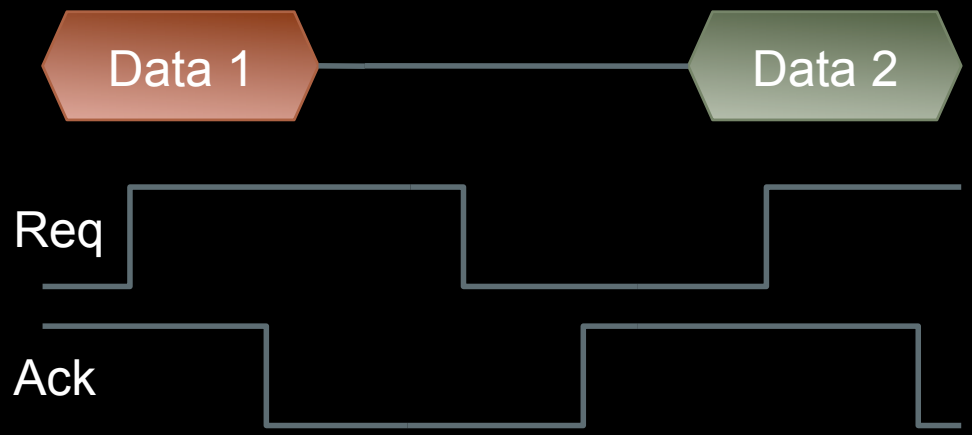
- 1ère Phase (Récepteur)
  - Détection de la requête
  - Consommation de la donnée
  - Acquittement
- 2ème Phase (Emetteur)
  - Détection de l'acquittement
  - Remise à zéro de la requête
- 3ème Phase (Récepteur)
  - Détection de la remise à zéro de la requête
  - Remise à zéro de l'acquittement
- 4ème Phase (Emetteur)
  - Détection de la remise à zéro de l'acquittement
  - Production d'une nouvelle donnée
  - Requête

jeremie.hamon@imag.fr

Concepts de base

Protocoles de communication : 4 phases (RZ)



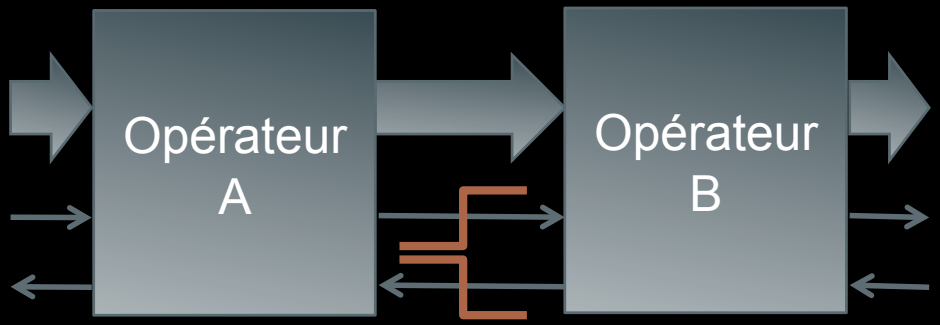


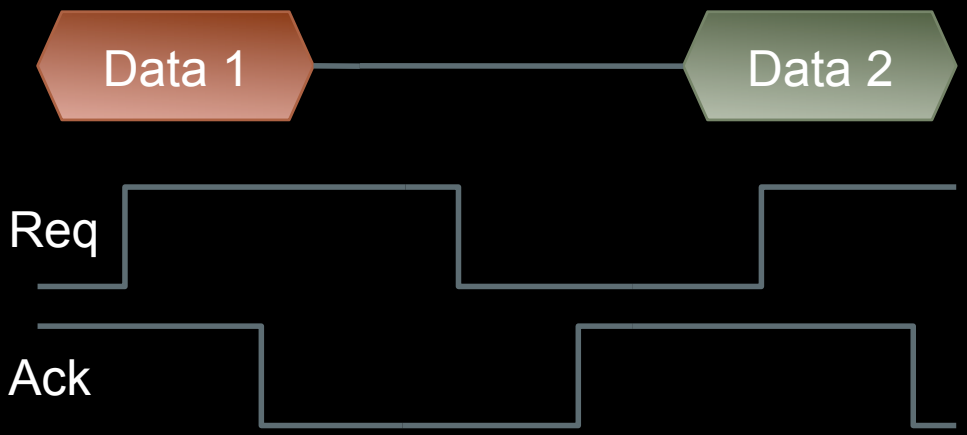
- 1ère Phase (Récepteur)
  - Détection de la requête
  - Consommation de la donnée
  - Acquittement
- 2ème Phase (Emetteur)
  - Détection de l'acquittement
  - Remise à zéro de la requête
- 3ème Phase (Récepteur)
  - Détection de la remise à zéro de la requête
  - Remise à zéro de l'acquittement
- 4ème Phase (Emetteur)
  - Détection de la remise à zéro de l'acquittement
  - Production d'une nouvelle donnée
  - Requête

jeremie.hamon@imag.fr

Concepts de base

Protocoles de communication : 4 phases (RZ)



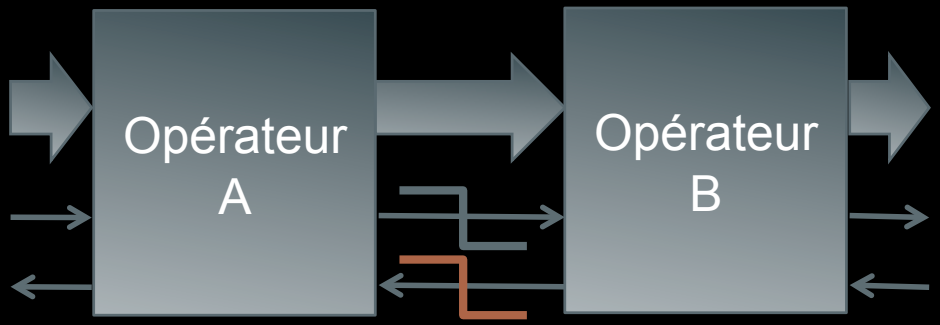


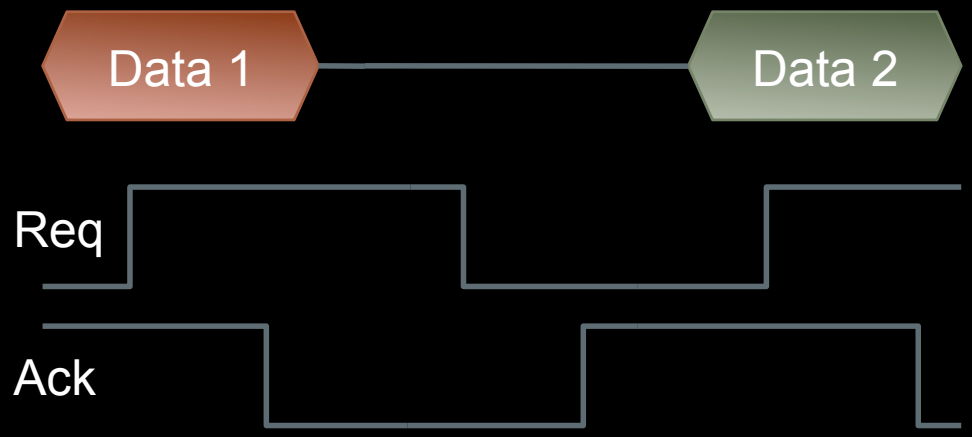
- 1ère Phase (Récepteur)
  - Détection de la requête
  - Consommation de la donnée
  - Acquittement
- 2ème Phase (Emetteur)
  - Détection de l'acquittement
  - Remise à zéro de la requête
- 3ème Phase (Récepteur)
  - Détection de la remise à zéro de la requête
  - Remise à zéro de l'acquittement
- 4ème Phase (Emetteur)
  - Détection de la remise à zéro de l'acquittement
  - Production d'une nouvelle donnée
  - Requête

jeremie.hamon@imag.fr

Concepts de base

Protocoles de communication : 4 phases (RZ)



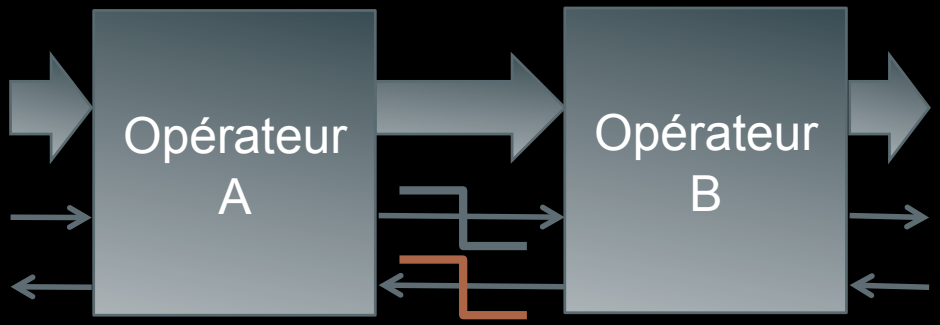


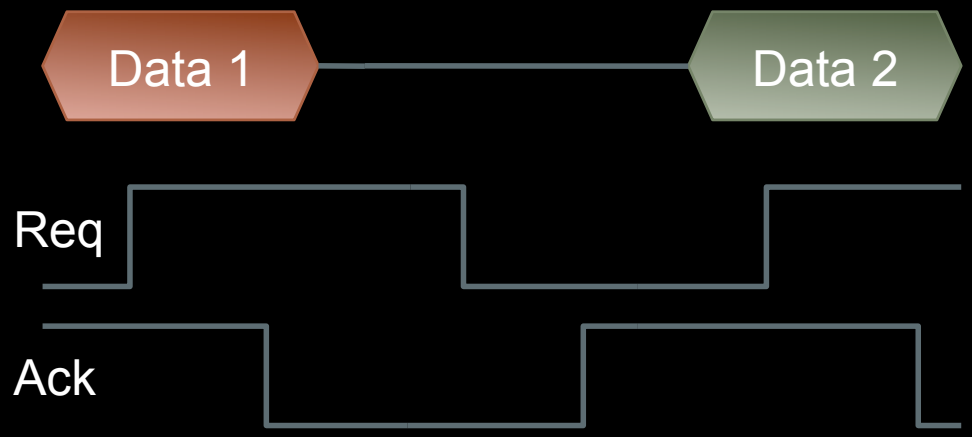
- 1ère Phase (Récepteur)
  - Détection de la requête
  - Consommation de la donnée
  - Acquittement
- 2ème Phase (Emetteur)
  - Détection de l'acquittement
  - Remise à zéro de la requête
- 3ème Phase (Récepteur)
  - Détection de la remise à zéro de la requête
  - Remise à zéro de l'acquittement
- 4ème Phase (Emetteur)
  - Détection de la remise à zéro de l'acquittement
  - Production d'une nouvelle donnée
  - Requête

jeremie.hamon@imag.fr

Concepts de base

Protocoles de communication : 4 phases (RZ)



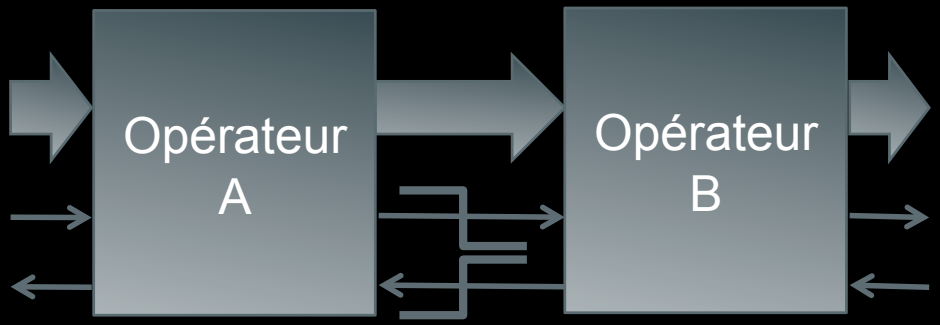


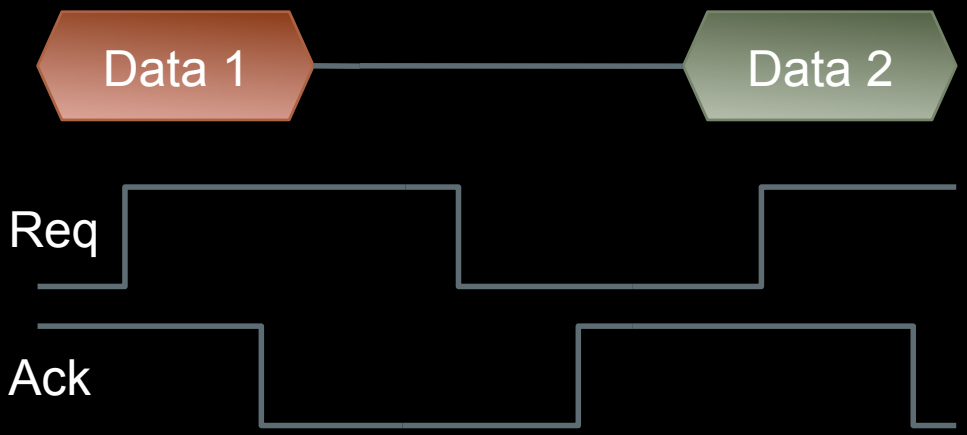
- 1ère Phase (Récepteur)
  - Détection de la requête
  - Consommation de la donnée
  - Acquittement
- 2ème Phase (Emetteur)
  - Détection de l'acquittement
  - Remise à zéro de la requête
- 3ème Phase (Récepteur)
  - Détection de la remise à zéro de la requête
  - Remise à zéro de l'acquittement
- 4ème Phase (Emetteur)
  - Détection de la remise à zéro de l'acquittement
  - Production d'une nouvelle donnée
  - Requête

jeremie.hamon@imag.fr

Concepts de base

Protocoles de communication : 4 phases (RZ)



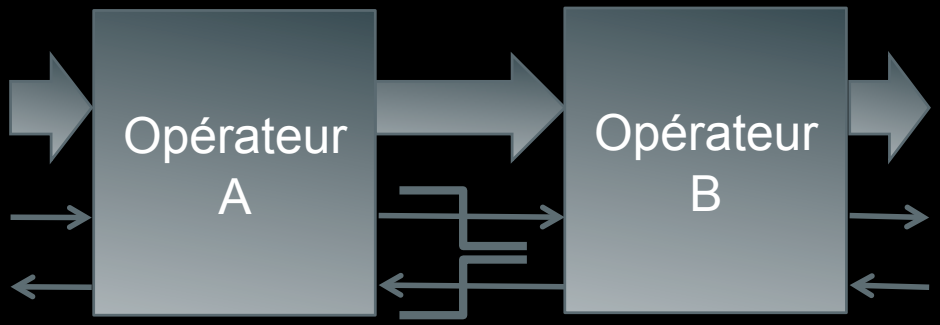


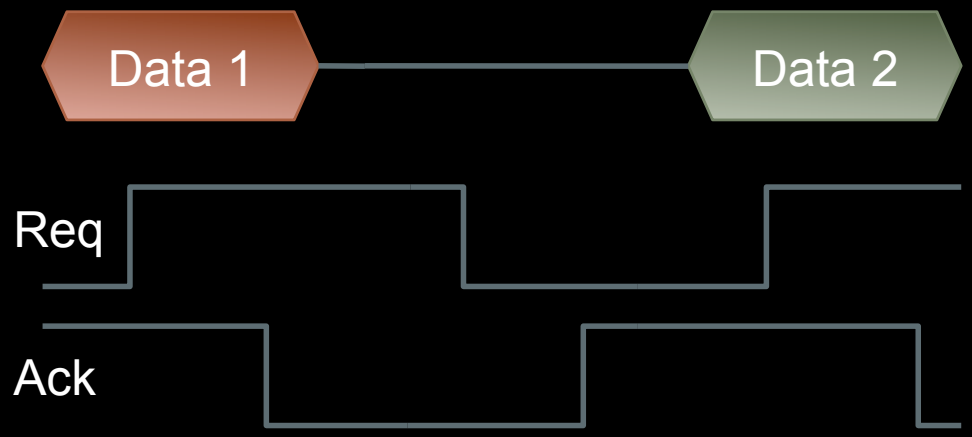
- 1ère Phase (Récepteur)
  - Détection de la requête
  - Consommation de la donnée
  - Acquittement
- 2ème Phase (Emetteur)
  - Détection de l'acquittement
  - Remise à zéro de la requête
- 3ème Phase (Récepteur)
  - Détection de la remise à zéro de la requête
  - Remise à zéro de l'acquittement
- 4ème Phase (Emetteur)
  - Détection de la remise à zéro de l'acquittement
  - Production d'une nouvelle donnée
  - Requête

jeremie.hamon@imag.fr

Concepts de base

Protocoles de communication : 4 phases (RZ)



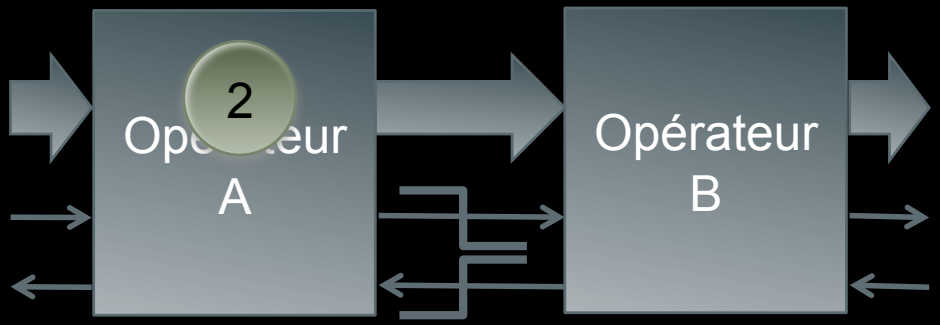


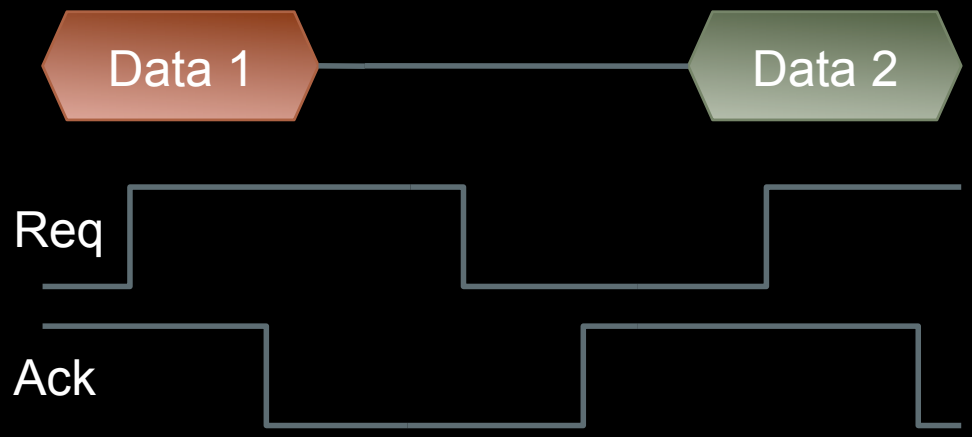
- 1ère Phase (Récepteur)
  - Détection de la requête
  - Consommation de la donnée
  - Acquittement
- 2ème Phase (Emetteur)
  - Détection de l'acquittement
  - Remise à zéro de la requête
- 3ème Phase (Récepteur)
  - Détection de la remise à zéro de la requête
  - Remise à zéro de l'acquittement
- 4ème Phase (Emetteur)
  - Détection de la remise à zéro de l'acquittement
  - Production d'une nouvelle donnée
  - Requête

jeremie.hamon@imag.fr

Concepts de base

Protocoles de communication : 4 phases (RZ)



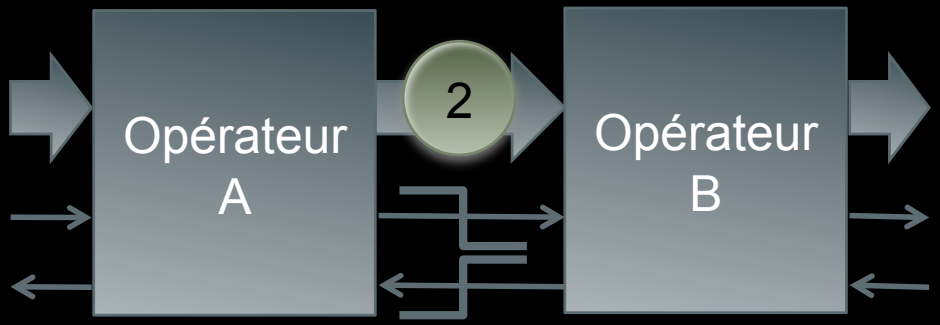


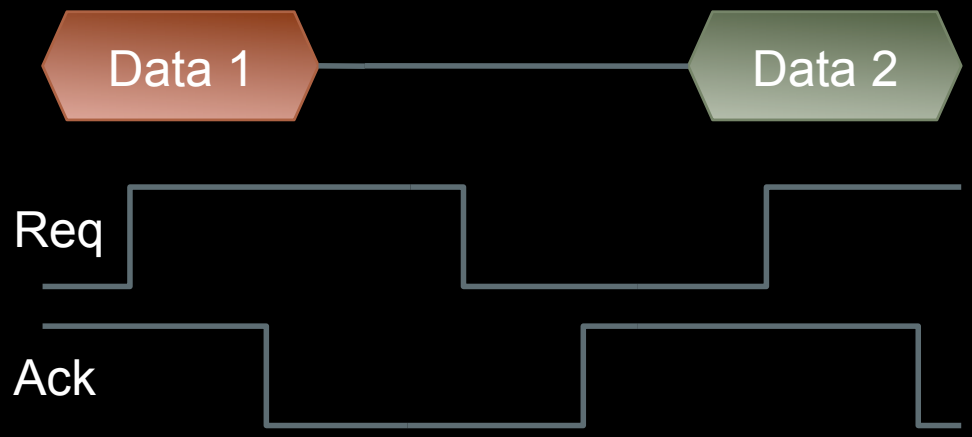
- 1ère Phase (Récepteur)
  - Détection de la requête
  - Consommation de la donnée
  - Acquittement
- 2ème Phase (Emetteur)
  - Détection de l'acquittement
  - Remise à zéro de la requête
- 3ème Phase (Récepteur)
  - Détection de la remise à zéro de la requête
  - Remise à zéro de l'acquittement
- 4ème Phase (Emetteur)
  - Détection de la remise à zéro de l'acquittement
  - Production d'une nouvelle donnée
  - Requête

jeremie.hamon@imag.fr

Concepts de base

Protocoles de communication : 4 phases (RZ)



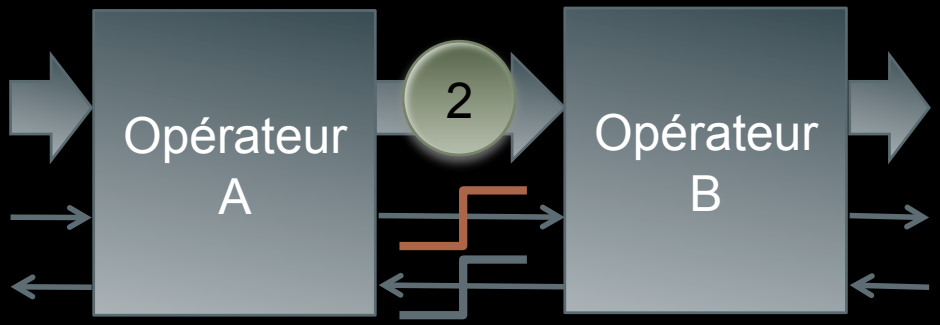


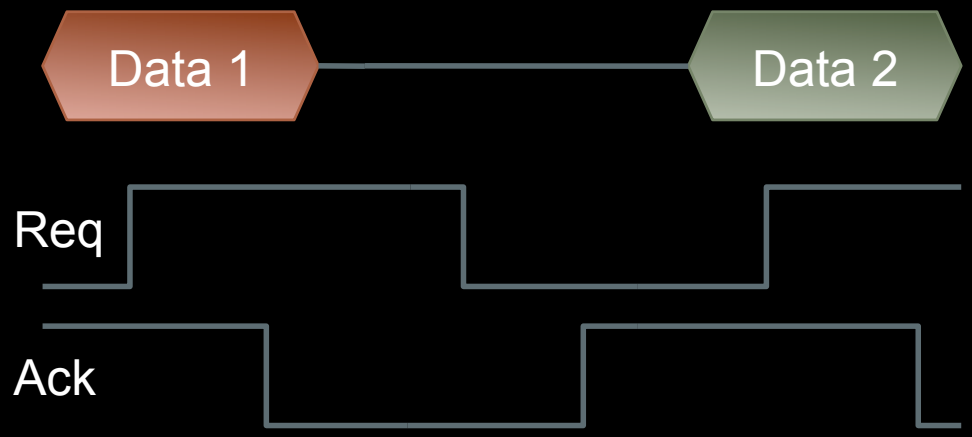
- 1ère Phase (Récepteur)
  - Détection de la requête
  - Consommation de la donnée
  - Acquittement
- 2ème Phase (Emetteur)
  - Détection de l'acquittement
  - Remise à zéro de la requête
- 3ème Phase (Récepteur)
  - Détection de la remise à zéro de la requête
  - Remise à zéro de l'acquittement
- 4ème Phase (Emetteur)
  - Détection de la remise à zéro de l'acquittement
  - Production d'une nouvelle donnée
  - Requête

jeremie.hamon@imag.fr

Concepts de base

Protocoles de communication : 4 phases (RZ)





- 1ère Phase (Récepteur)
  - Détection de la requête
  - Consommation de la donnée
  - Acquittement
- 2ème Phase (Emetteur)
  - Détection de l'acquittement
  - Remise à zéro de la requête
- 3ème Phase (Récepteur)
  - Détection de la remise à zéro de la requête
  - Remise à zéro de l'acquittement
- 4ème Phase (Emetteur)
  - Détection de la remise à zéro de l'acquittement
  - Production d'une nouvelle donnée
  - Requête

jeremie.hamon@imag.fr

Concepts de base

Protocoles de communication

2 phases

- Moins de transitions
- Logique à front (complexe)
- Optimisation du pipeline difficile
- Réservé généralement aux blocs qui présentent des latences élevés (plots du circuit par exemple)

4 phases

- Plus de transitions
- Logique à niveau (simple)
- Possibilité de masquer la pénalité des phases de remise à zéro par optimisation du pipeline asynchrone
- Utilisé généralement pour implémenter le cœur du circuit

Concepts de base

Codage des données

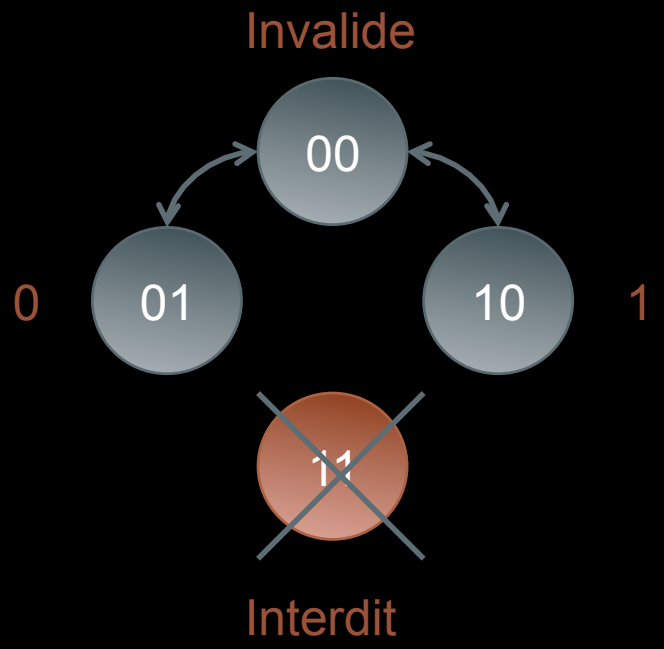
- Codage données groupées (Bundle Data)
  - 1 fil par bit de donnée
  - 1 fil supplémentaire pour la validité (requête)
  - Simplicité / Compacité ☺
  - Codage sensible aux délais ☹
    - La requête ne doit pas précéder la validité effective des données.
    - Le temps d'exécution ne dépend pas des données traitées.

Concepts de base

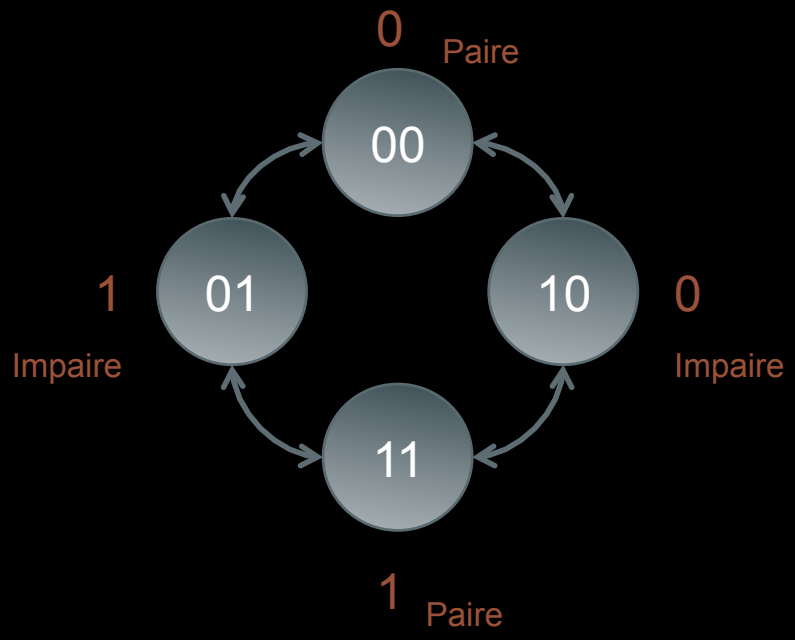
Codage des données

- Codages insensibles aux délais (double-rail) :
  Validité encapsulée dans les données ➔ 2 fils (rails) par bit de donnée

- Codage 3 états :



- Codage 4 états :

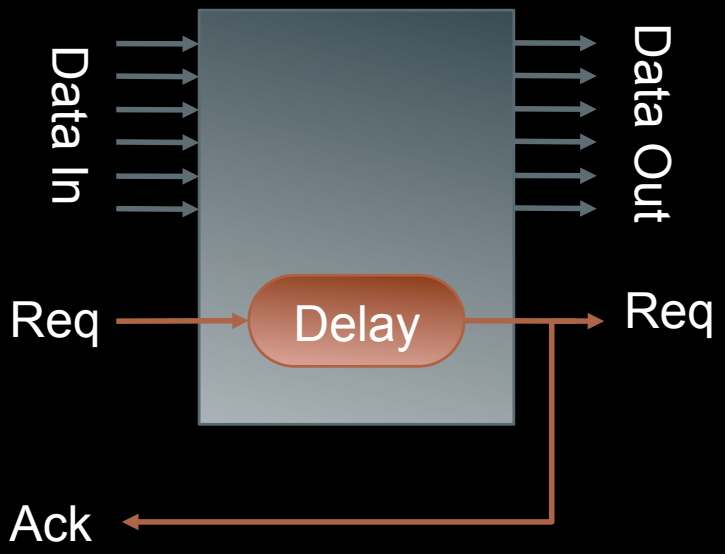


jeremie.hamon@imag.fr

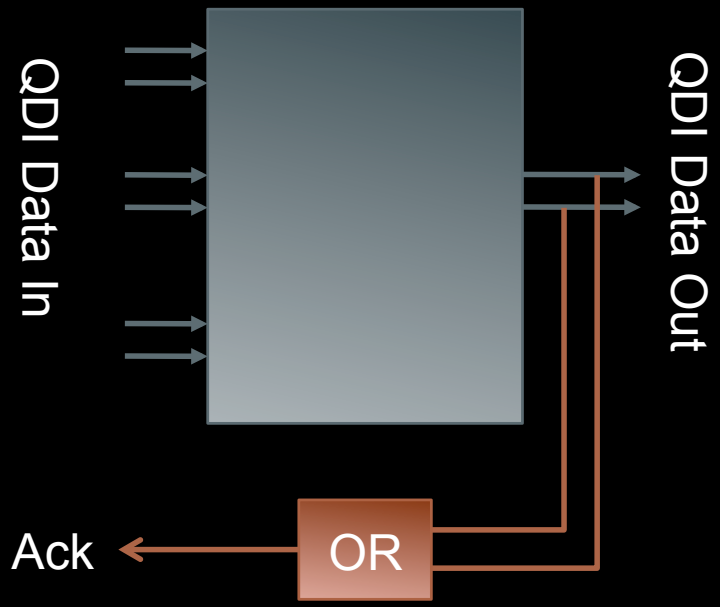
Concepts de base

Signaux d'acquittement

- Codage « Bundle Data »



- Codage QDI (3 états)



jeremie.hamon@imag.fr

Première partie

Introduction à la logique asynchrone

- Approche synchrone : Intérêt… et limitations
- Concepts de base de la logique asynchrone
  - Synchronisation locale
  - Protocoles de communication
  - Codage des données
  - Signaux d'acquittement
- Classe des circuits asynchrones
- Propriétés
  - Absence d'horloge
  - Calcul en temps minimal
  - Technologies déca-nanométrique
  - Modularité
  - Faible consommation
  - Faible EMI

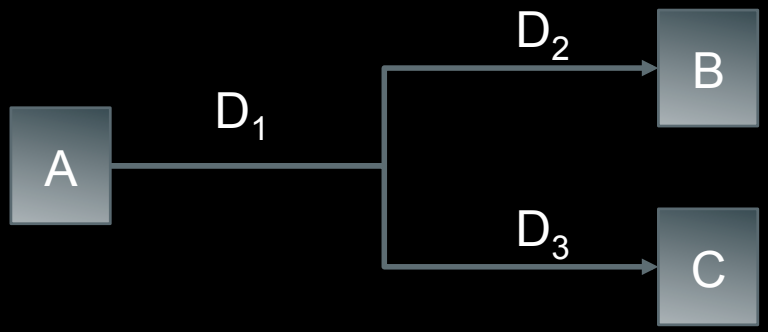
jeremie.hamon@imag.fr

Classe des circuits asynchrones

Modèles de délais

- Circuits classifiés en fonction des hypothèses temporelles effectuées pendant la conception
- Modèles de délais pour les portes et les interconnexions
  - Délai fixe : valeur connue précisément
  - Délai borné : valeur comprise dans un intervalle connu précisément
  - Délai non-borné : valeur finie mais non connue
- Fourche isochrone :
  - Délai non-borné
  - Délai égal sur les branches

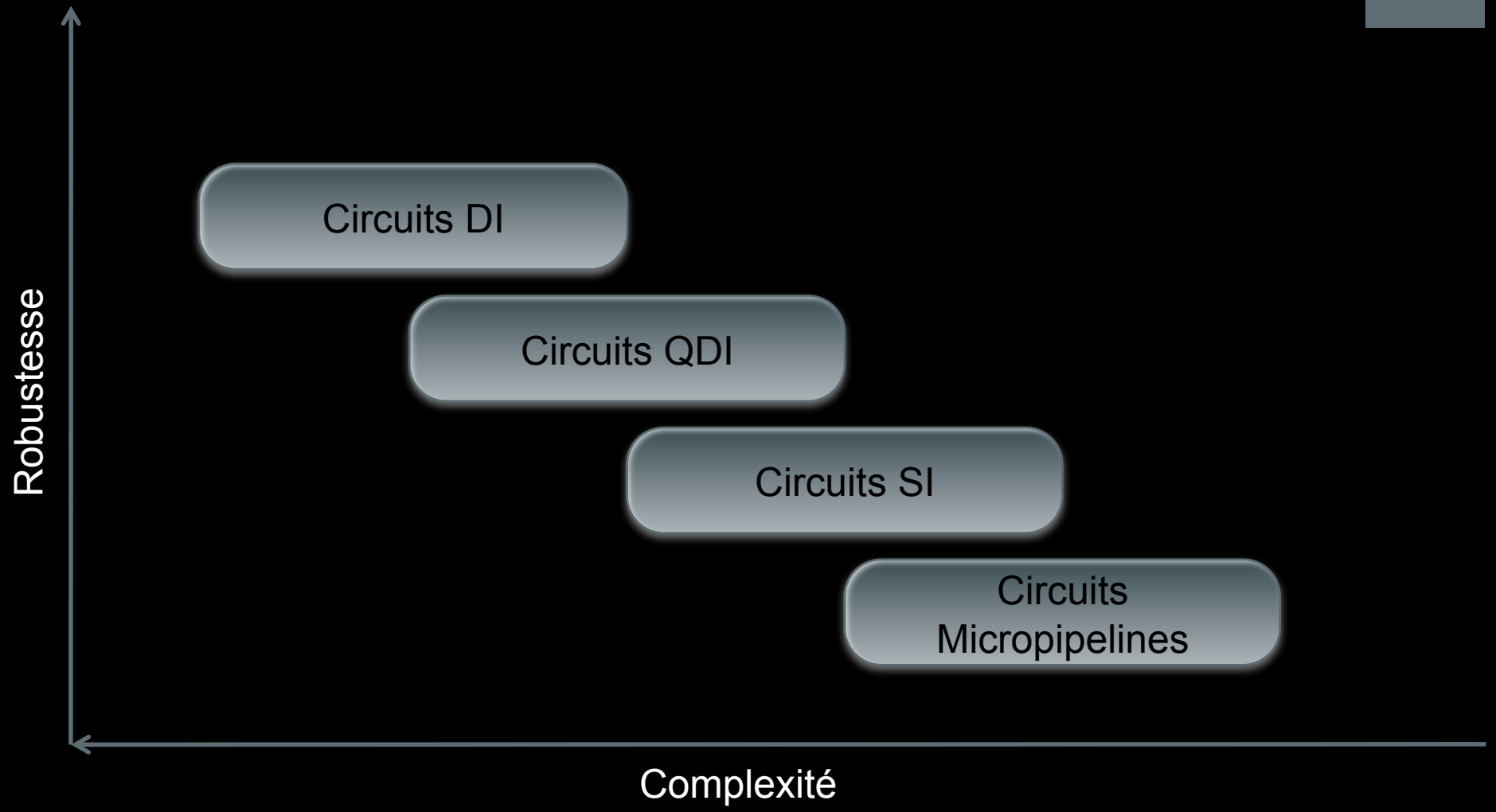
  de la fourche : $D_1+D_2 = D_1+D_3$



jeremie.hamon@imag.fr

Classe des circuits asynchrones

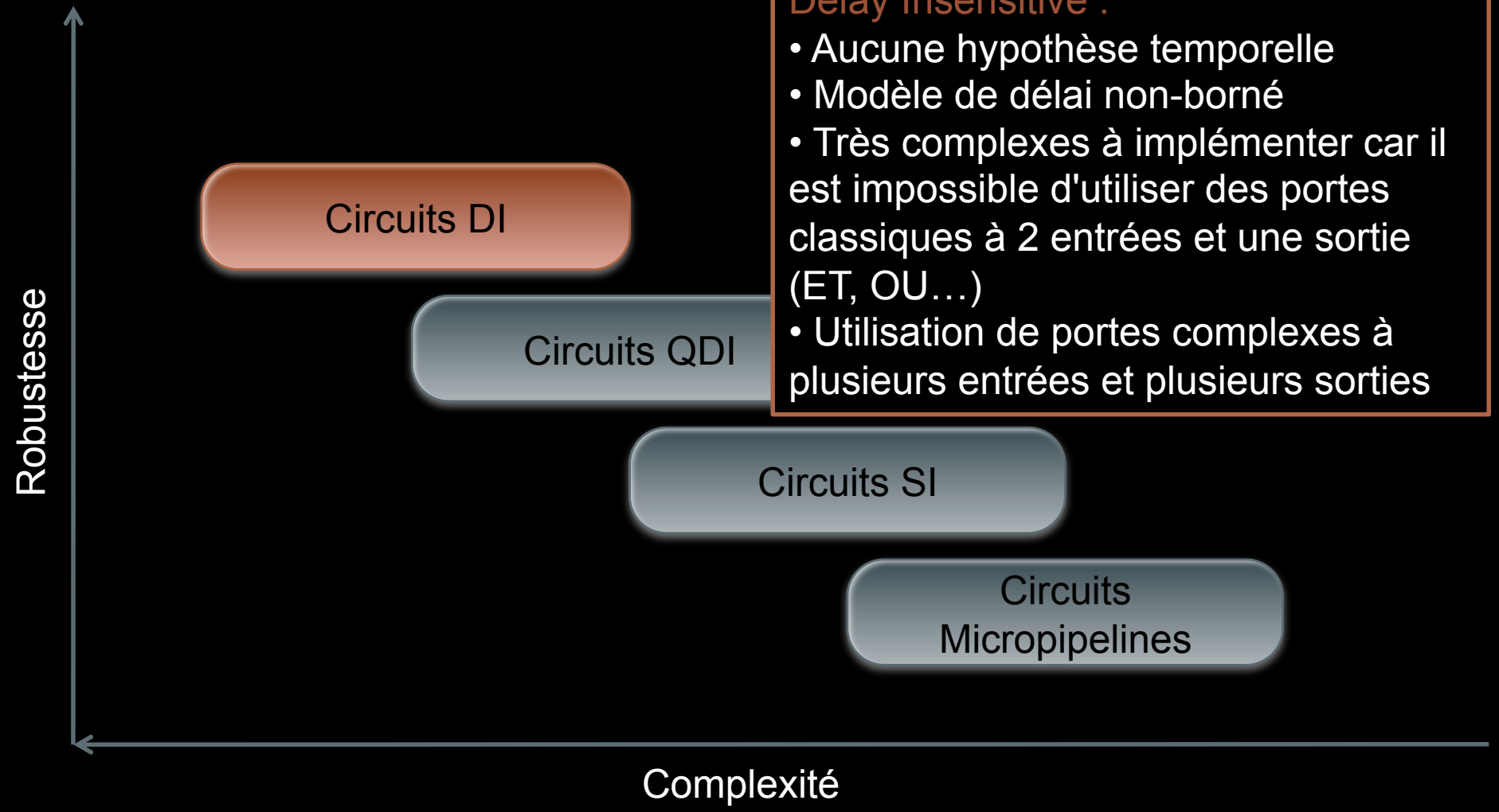
Robustesse vs Complexité



jeremie.hamon@imag.fr

Classe des circuits asynchrones

Robustesse vs Complexité



Delay Insensitive :

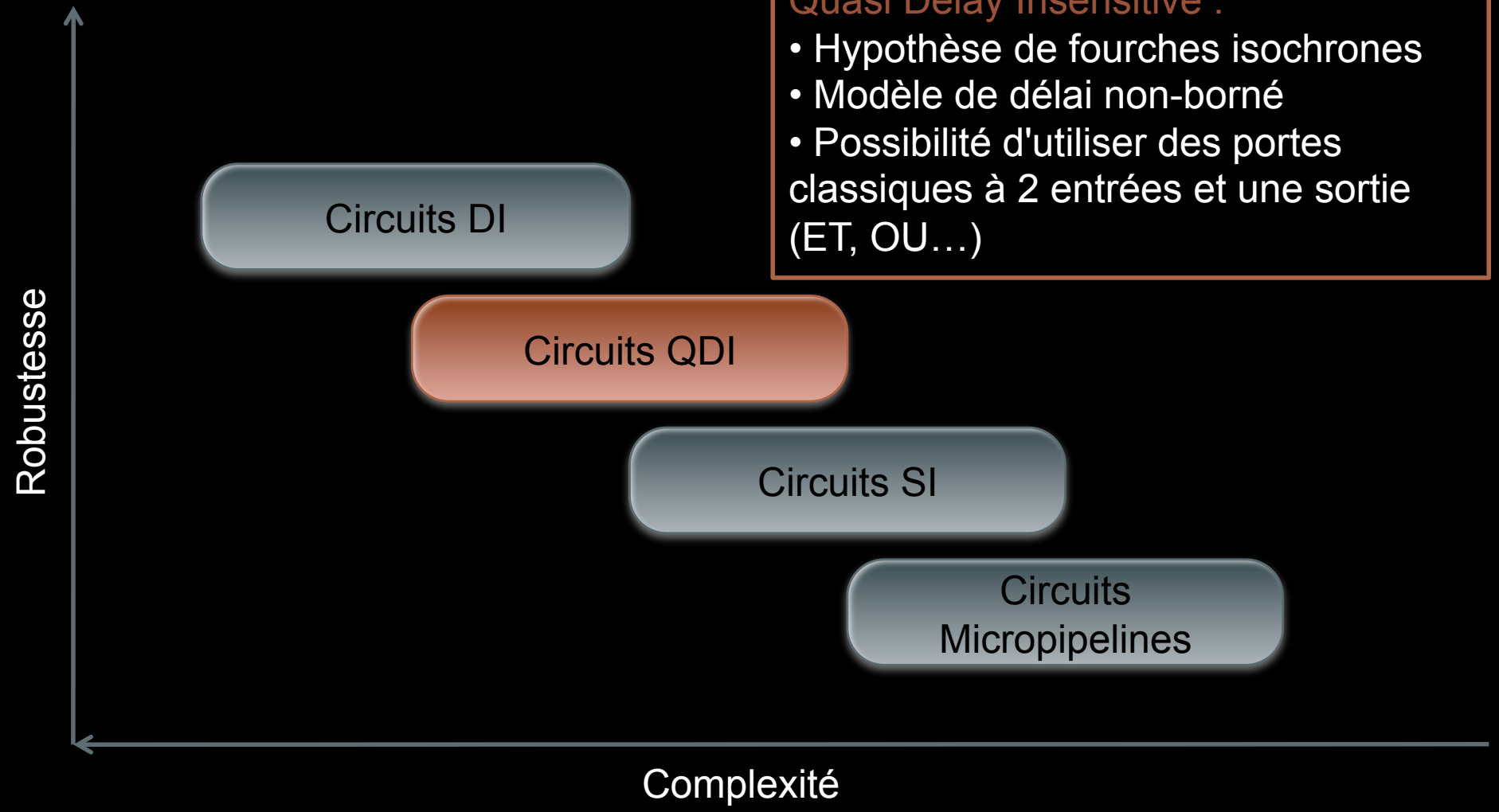
- Aucune hypothèse temporelle
- Modèle de délai non-borné
- Très complexes à implémenter car il est impossible d'utiliser des portes classiques à 2 entrées et une sortie (ET, OU…)
- Utilisation de portes complexes à plusieurs entrées et plusieurs sorties

jeremie.hamon@imag.fr

Classe des circuits asynchrones

Robustesse vs Complexité

Quasi Delay Insensitive :
- Hypothèse de fourches isochrones
- Modèle de délai non-borné
- Possibilité d'utiliser des portes classiques à 2 entrées et une sortie (ET, OU…)

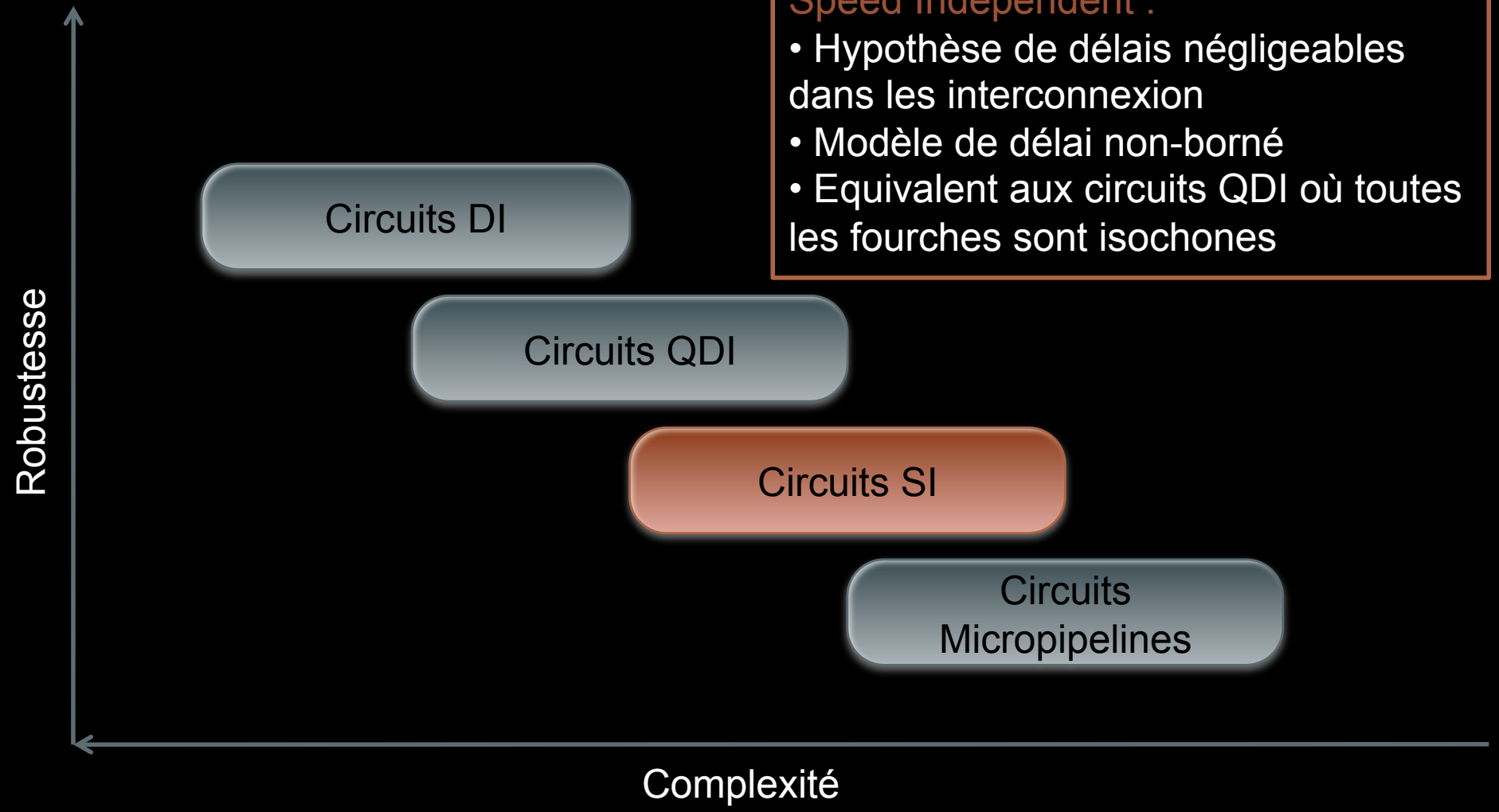


jeremie.hamon@imag.fr

Classe des circuits asynchrones

Robustesse vs Complexité

Speed Independent :
- Hypothèse de délais négligeables dans les interconnexion
- Modèle de délai non-borné
- Equivalent aux circuits QDI où toutes les fourches sont isochones

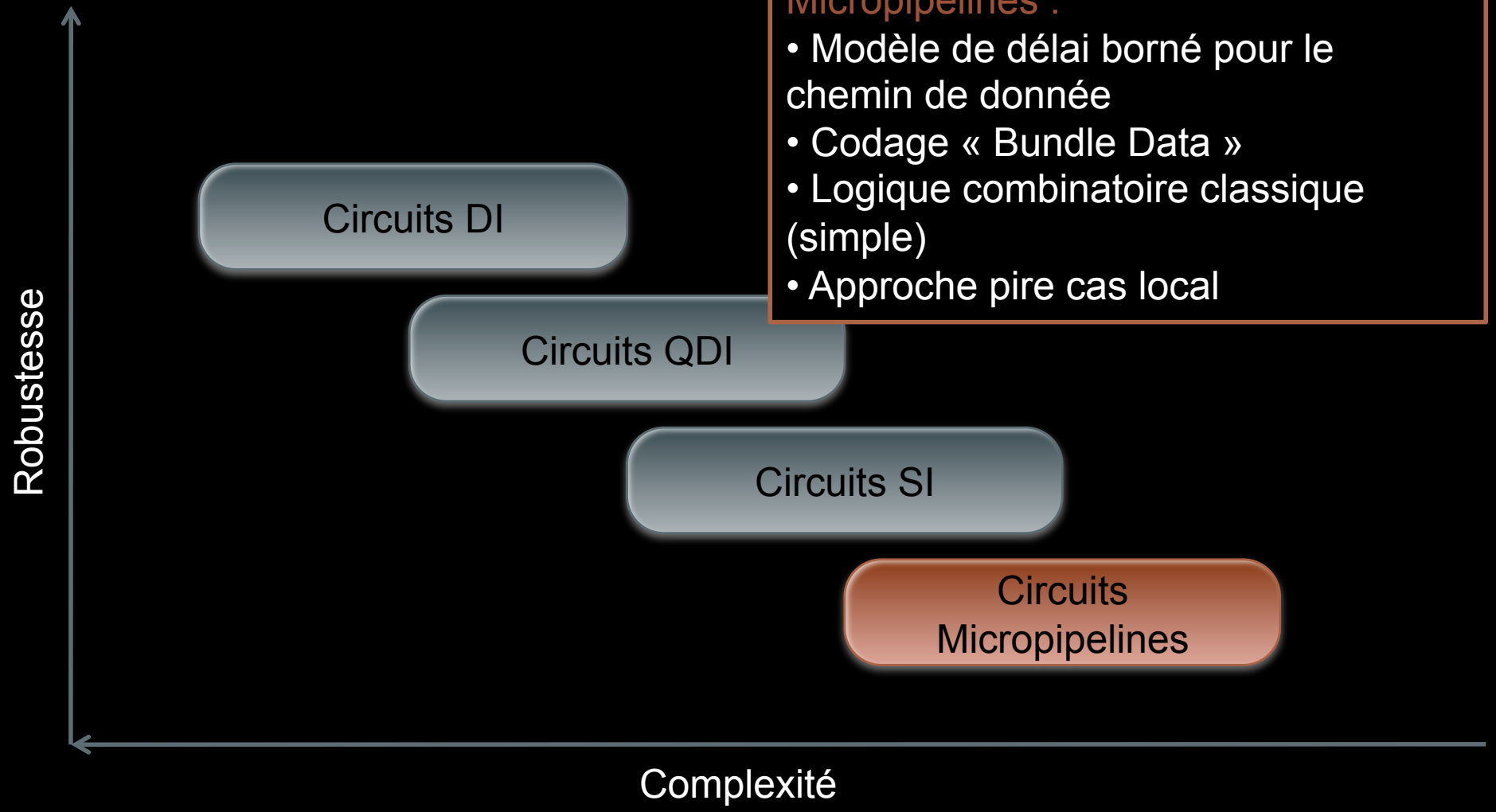


jeremie.hamon@imag.fr

Classe des circuits asynchrones

Robustesse vs Complexité

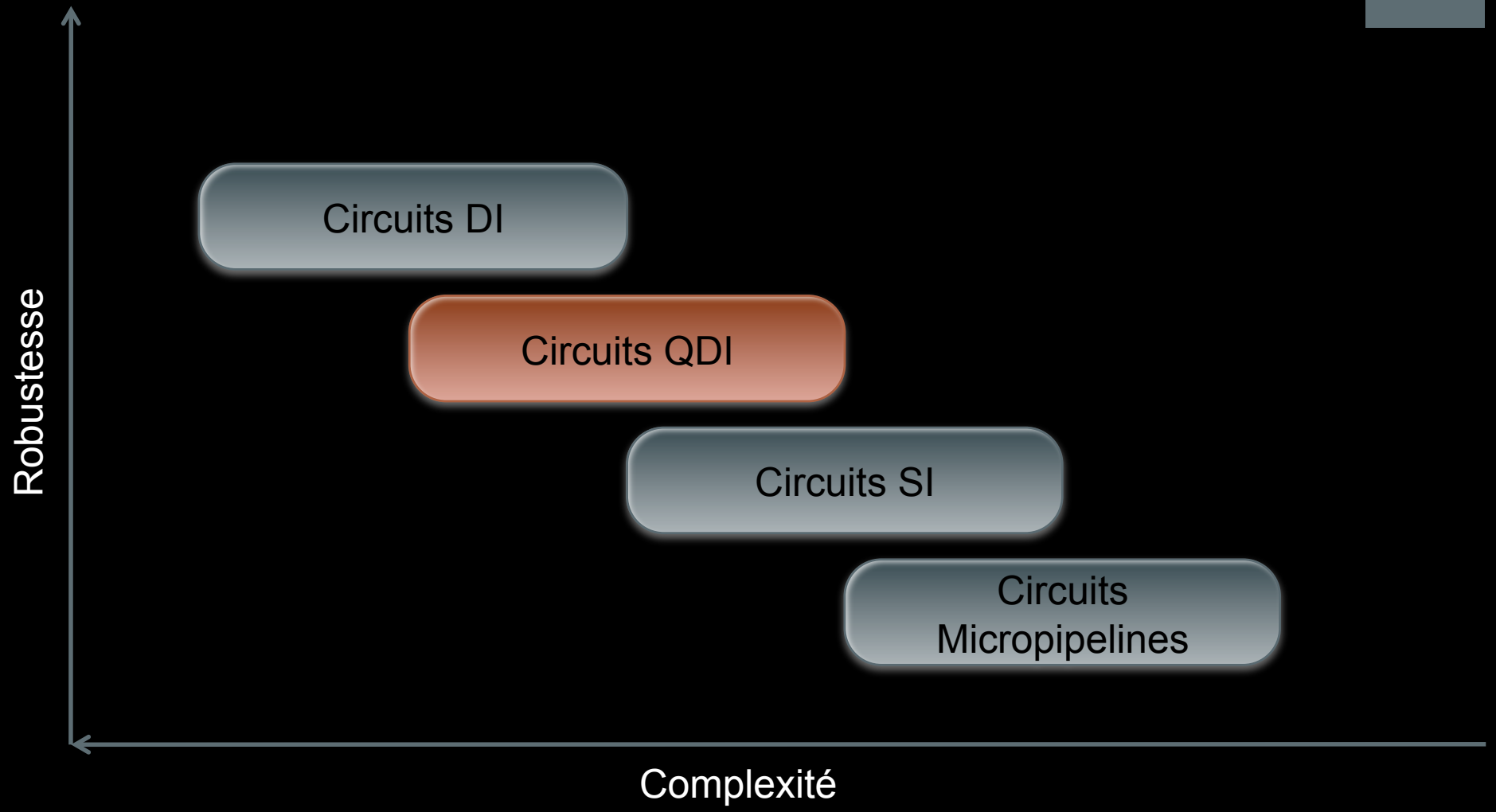
Micropipelines :
- Modèle de délai borné pour le chemin de donnée
- Codage « Bundle Data »
- Logique combinatoire classique (simple)
- Approche pire cas local



jeremie.hamon@imag.fr

Classe des circuits asynchrones

Robustesse vs Complexité



jeremie.hamon@imag.fr

Première partie

Introduction à la logique asynchrone

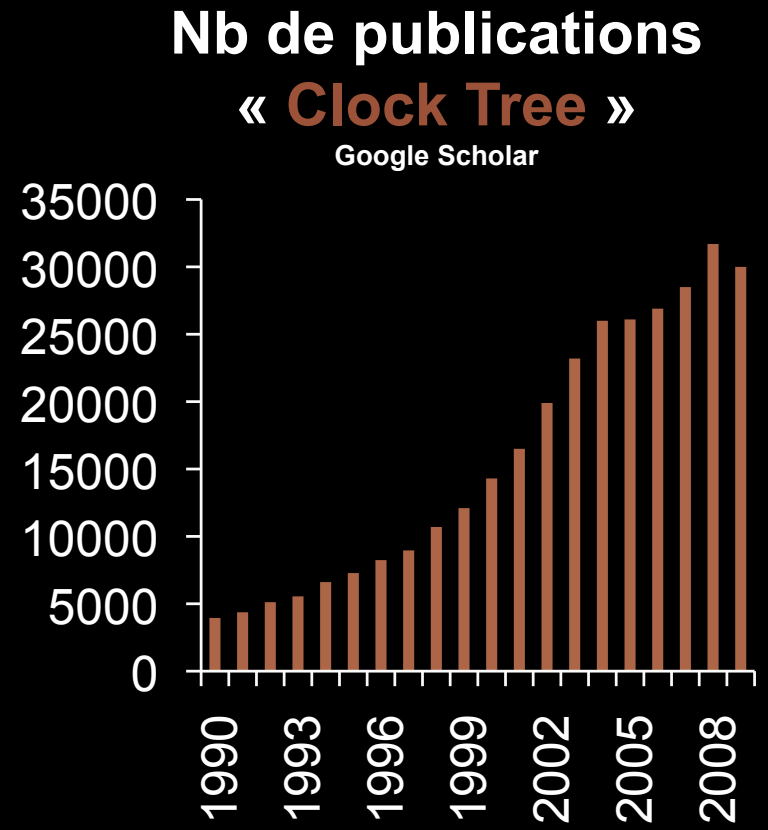
- Approche synchrone : Intérêt… et limitations
- Concepts de base de la logique asynchrone
  - Synchronisation locale
  - Protocoles de communication
  - Codage des données
  - Signaux d'acquittement
- Classe des circuits asynchrones
- Propriétés
  - Absence d'horloge
  - Calcul en temps minimal
  - Technologies déca-nanométrique
  - Modularité
  - Faible consommation
  - Faible EMI

jeremie.hamon@imag.fr

Propriétés des circuits asynchrones

Absence d'horloge

- Génération du signal d'horloge :
  - PLL, DLL
  - Jitter
  - Consommation
- Distribution du signal d'horloge :
  - Complexité
  - Skew
  - Consommation
- Caractérisation/Estimation des délais :
  - Cellules élémentaires
  - Interconnexions

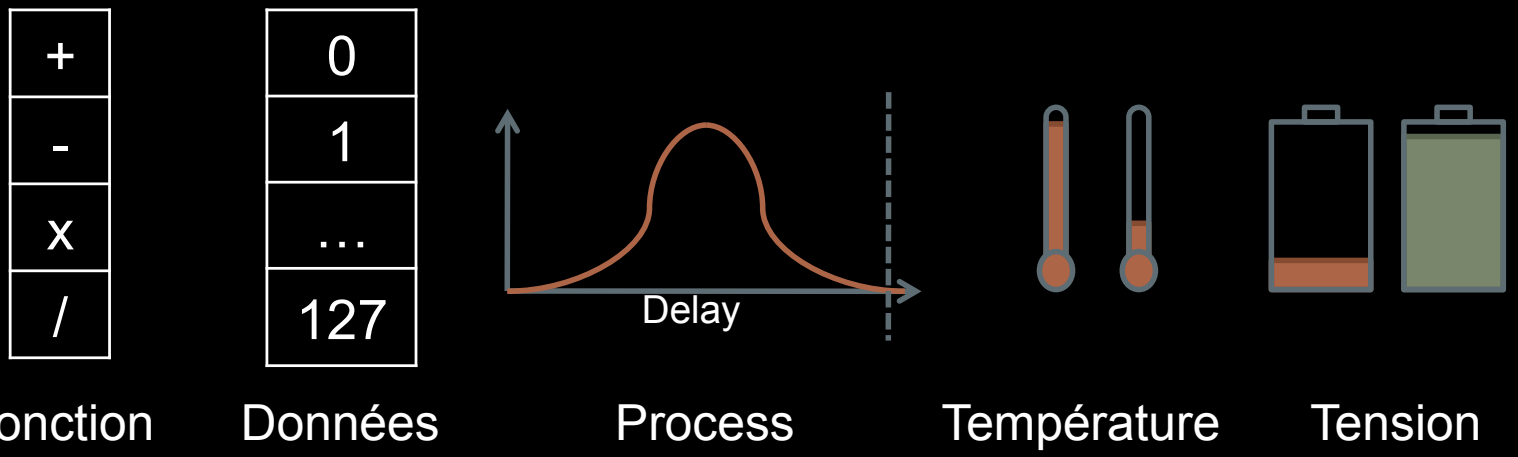


jeremie.hamon@imag.fr

Propriétés des circuits asynchrones

Calcul en temps minimum

- Variabilité du temps de traitement d'un opérateur logique/ arithmétique :



- La synchronisation locale implémentée par tous les éléments du système permet de tirer parti de la variabilité des temps de traitements : Vitesse maximale !!!

jeremie.hamon@imag.fr

Propriétés des circuits asynchrones

Technologies déca-nanométriques

- Performances ?
- Rendement ?
- ➔ Design for Variability (DFV)
  - Modélisation complexe
  - Conception complexe
  - …

Technology (L Drawn)	Data	VDD	W	T_{OXE}	Mean V_T	Sigma V_T
340 nm	Foundry	3.3 V	360 nm	7.2 nm	439 mV	18 mV
240 nm	Foundry	2.5 V	360 nm	6.0 nm	397 mV	21 mV
180 nm	Foundry	1.8 V	240 nm	3.90 nm	366 mV	18 mV
90 nm	Foundry	1.2 V	160 nm	2.95 nm	409 mV	31 mV
80 nm	Foundry	1.2 V	120 nm	2.25 nm	300 mV	27 mV
60 nm	Measurement [5]	1.2 V	140 nm	2 nm	--	29 mV
45 nm	Measurement [6]	1.1 V	--	--	--	45 mV
25 nm *(UTB-SOI)*	Measurement [2]	1.0 V	60 nm	1.65 nm	480 mV	25 mV
35 nm	Simulation [3]	0.85 V	--	0.88 nm	226 mV	30 mV
13 nm	Simulation [3]	0.85 V	--	0.44 nm	226 mV	82 mV

La technologie asynchrone est intrinsèquement robuste aux variations PVT

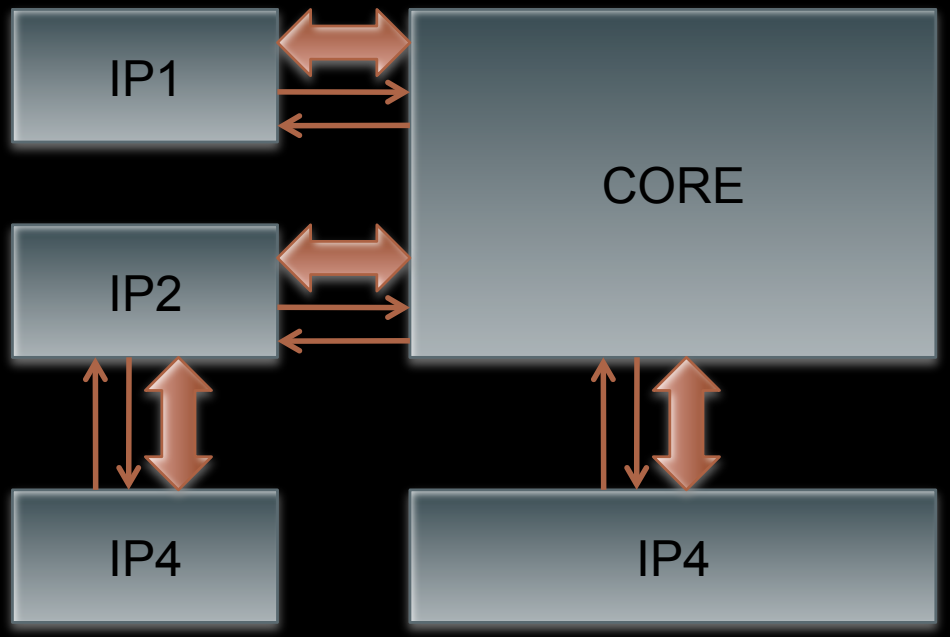
[PATMOS 2010] : Kheradmand-Boroujeni, Christian Piguet and Yusuf Leblebici, « Logic Architecture and VDD Selection for Reducing the Impact of Intra-die Random VT Variations on Timing »

jeremie.hamon@imag.fr

Propriétés des circuits asynchrones

Modularité

- Conception de systèmes complexes par assemblage (Lego)
- « Design and Reuse » / « IP »
- Optimisation / modification de certains blocs

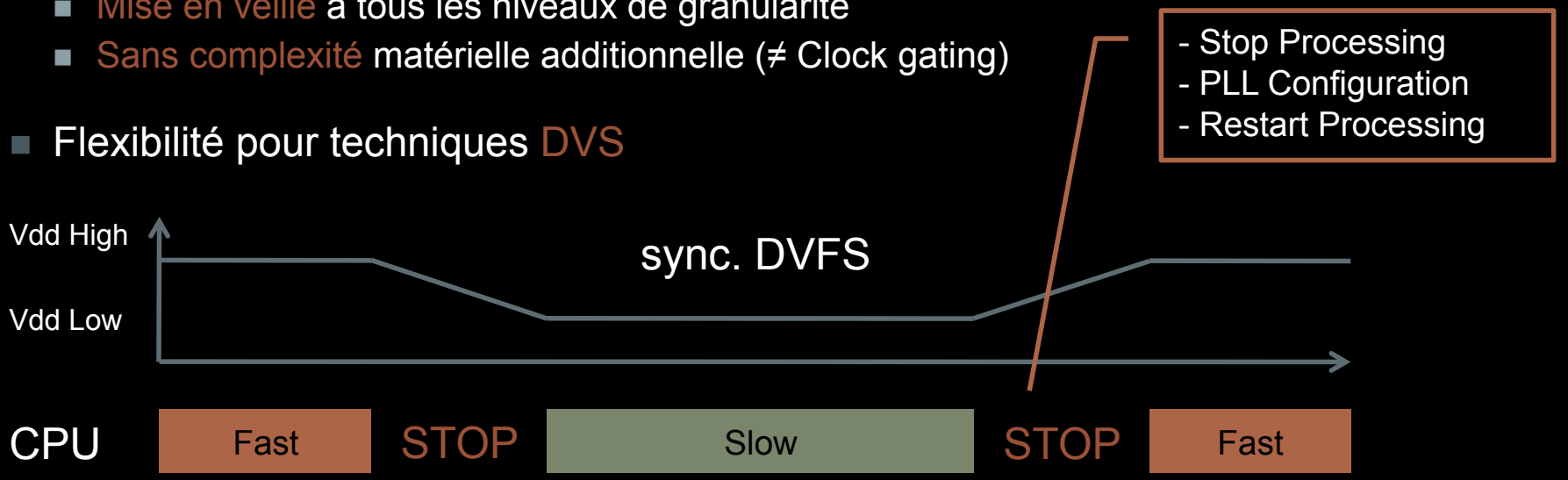


jeremie.hamon@imag.fr

Propriétés des circuits asynchrones

Faible consommation

- Absence d'horloge :
  Génération + Distribution + Mémorisation > 50 % conso totale
- Conception sans aléas
- Logique « Data Flow » :
  - Seuls les blocs impliqués dans un traitement consomment
  - Mise en veille à tous les niveaux de granularité
  - Sans complexité matérielle additionnelle (≠ Clock gating)
- Flexibilité pour techniques DVS



jeremie.hamon@imag.fr

Propriétés des circuits asynchrones

Faible consommation

- Absence d'horloge :
  Génération + Distribution + Mémorisation > 50 % conso totale
- Conception sans aléas
- Logique « Data Flow » :
  - Seuls les blocs impliqués dans un traitement consomment
  - Mise en veille à tous les niveaux de granularité
  - Sans complexité matérielle additionnelle (≠ Clock gating)
- Flexibilité pour techniques DVS

jeremie.hamon@imag.fr

Propriétés des circuits asynchrones

Faible consommation – exemples : Low Power CPU

- Texas Instruments TI MSP 430
  - 16 bits
  - 16 MHz
  - 200 uA/MIPS
  - Wake-Up time ~ 1 us
- Handshake Solutions HT-80C51
  - 8 bits
  - Micropipeline (6,3 MIPS)
  - 80 uA/MIPS
  - Wake-Up time ?
- Silicon Lab C8051F9xx
  - 8 bits
  - 25 MHz
  - 160 uA/MIPS
  - Wake-Up time ~ 2 us
- Tiempo TAM16
  - 16 bits
  - QDI (7~15 MIPS)
  - < 50 uA/MIPS
  - Wake-Up time ~ 5 ns

jeremie.hamon@imag.fr

Propriétés des circuits asynchrones

Faible bruit - EMI

- Synchrone :
  - Tous les traitements débutent aux mêmes instants (front d'horloge)
  - Forts appels de courant à la fréquence de l'horloge

➔ Rayonnement électromagnétique aux harmoniques de la fréquence d'horloge

- Asynchrone :
  - Fonctionnement « flot de données »
  - Traitements distribués dans le temps
  - Appels de courant limités

➔ Diminution du rayonnement électromagnétique

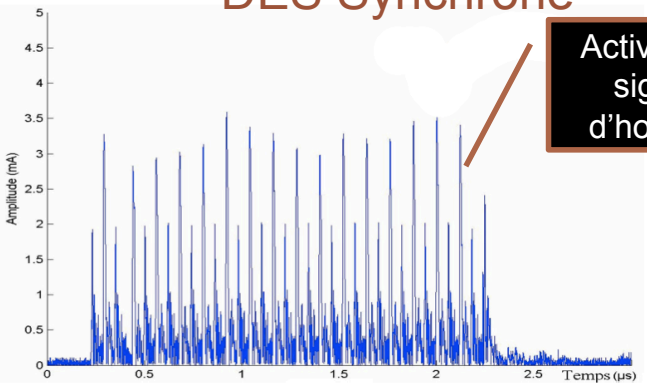
jeremie.hamon@imag.fr

Propriétés des circuits asynchrones

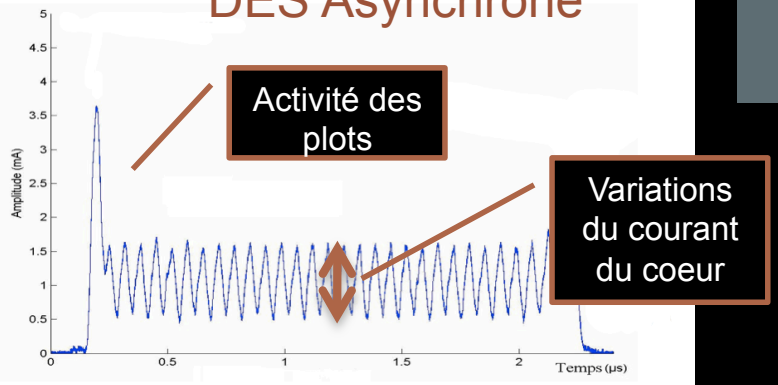
Profiles de courant

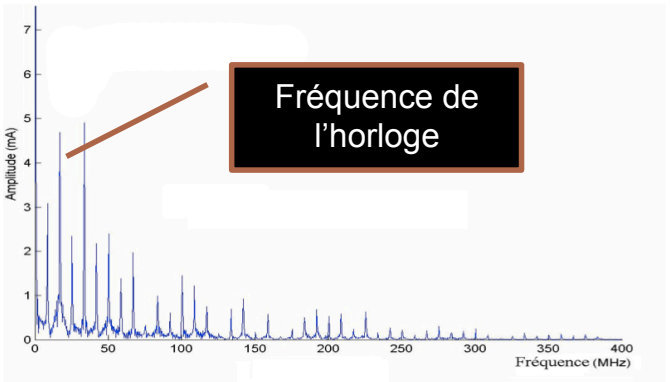
Spectres de courant

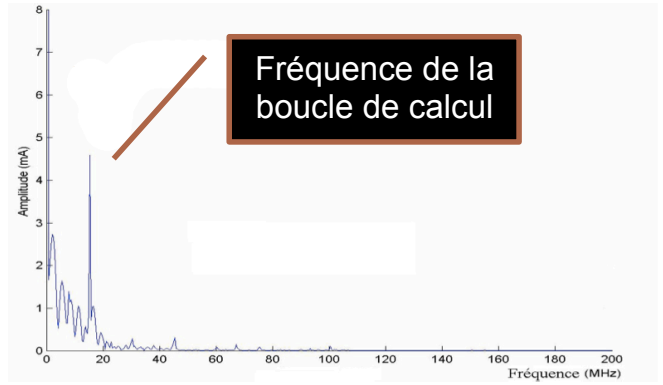
DES Synchrone



DES Asynchrone







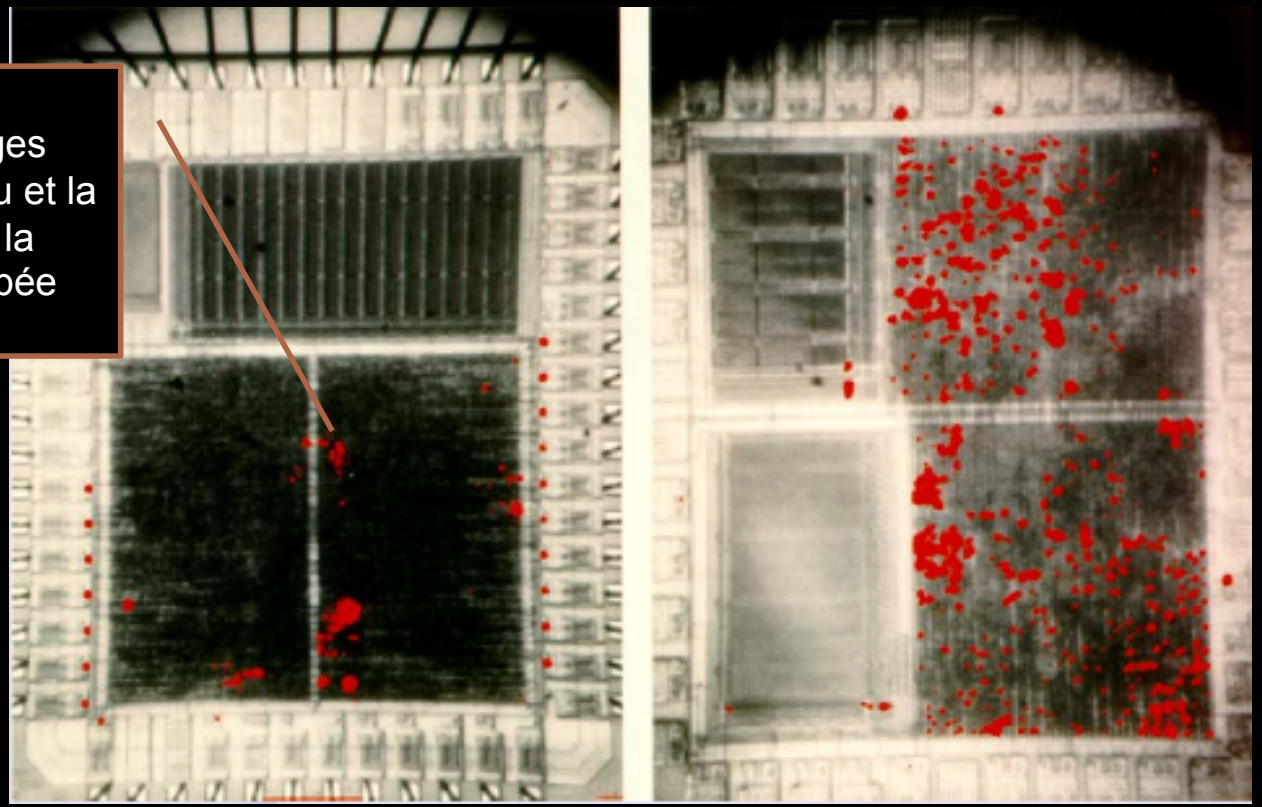
[BOUESSE 2005] : Ghislain Fraidy Bouesse, « Contribution à la conception de circuits intégrés : l'alternative asynchrone », Thèse, Grenoble-INP

jeremie.hamon@imag.fr

Propriétés des circuits asynchrones

Faible bruit - EMI

Les points rouges indiquent le niveau et la distribution de la puissance dissipée



[ASYNC 2006] : Rik van de Wiel, Handshake Solutions, Async Industrial Session

Handshake 80C51 Clocked 80C51

Deuxième partie

Introduction à la conception en logique asynchrone

- Conception sans aléas
- La porte de Muller
- Synthèse de la logique QDI
  - Logique DIMS
- Eléments de mémorisation
  - Le Half-Buffer
- Architecture des circuits asynchrones
  - Pipeline asynchrone linéaire
  - Pipeline asynchrone non-linéaire
  - Deadlocks

jeremie.hamon@imag.fr

Deuxième partie

Introduction à la conception en logique asynchrone

- Conception sans aléas
- La porte de Muller
- Synthèse de la logique QDI
  - Logique DIMS
- Eléments de mémorisation
  - Le Half-Buffer
- Architecture des circuits asynchrones
  - Pipeline asynchrone linéaire
  - Pipeline asynchrone non-linéaire
  - Deadlocks

Conception sans aléa

Introduction

- Exemples :

- Aléas fonctionnels : possibilité de les éviter lors de la spécification
- Aléas combinatoires : nécessité d'une implémentation exempte d'aléa
  - Aléas de type SIC (Single Input Change)
  - Aléas de type MIC (Multiple Input Change)

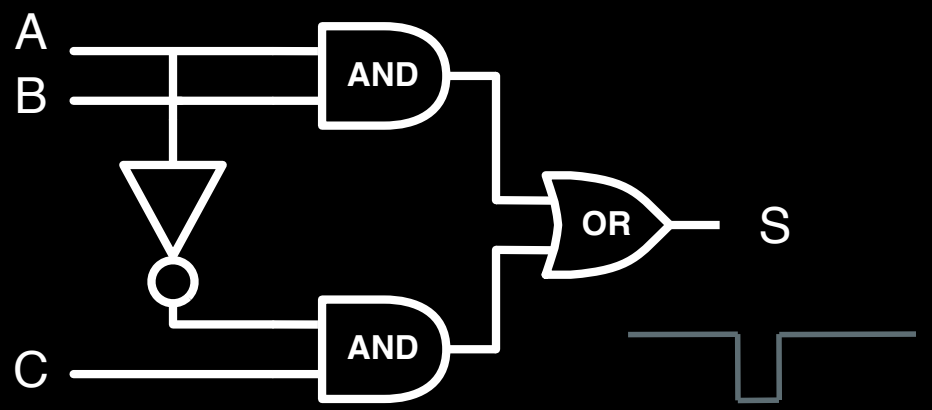
Conception sans aléa

Exemple : Single Input Change

- Tableau de Karnaugh

cb \ a	0	1
00	0	0
01	0	1
11	1	1
10	1	0

$s = a.b + \neg a.c$



jeremie.hamon@imag.fr

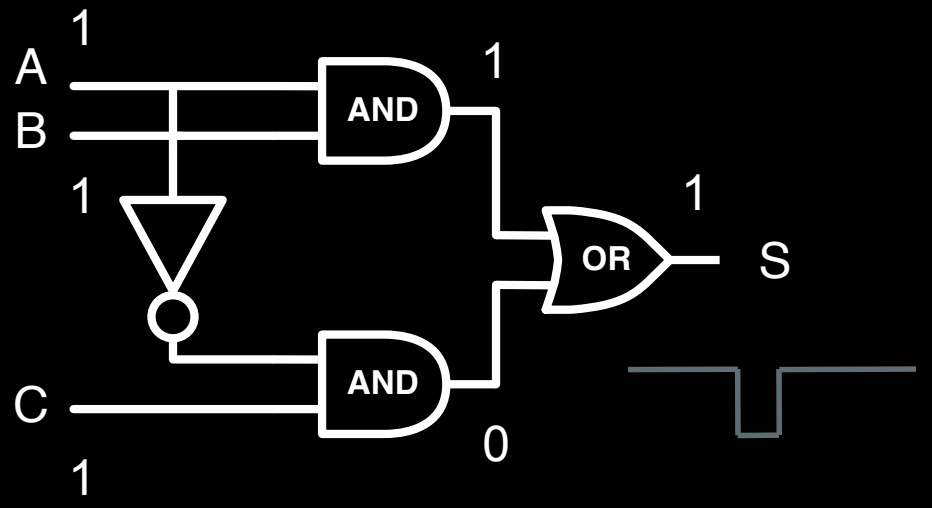
Conception sans aléa

Exemple : Single Input Change

- Tableau de Karnaugh

cb \ a	0	1
00	0	0
01	0	1
11	1	1
10	1	0

$s = a.c + \neg a.b$



jeremie.hamon@imag.fr

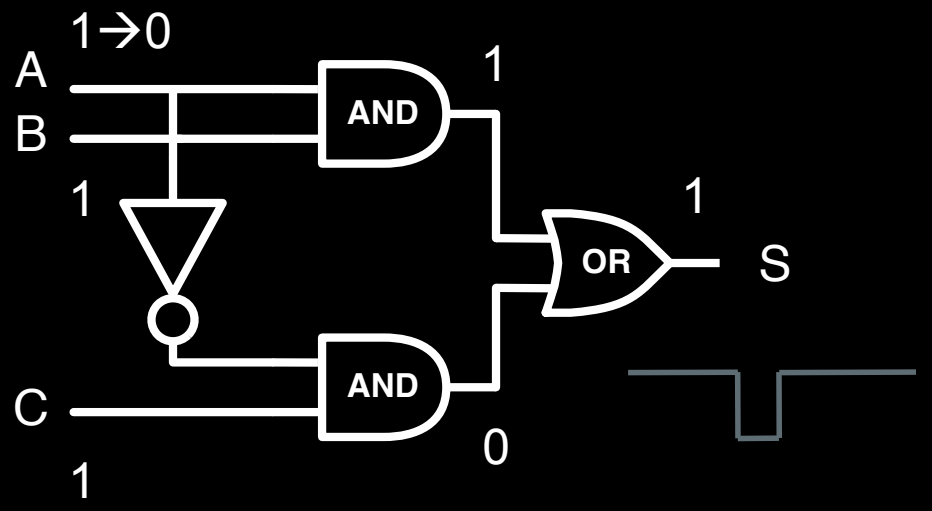
Conception sans aléa

Exemple : Single Input Change

- Tableau de Karnaugh

cb \ a	0	1
00	0	0
01	0	1
11	1	1
10	1	0

$s = a.c + \neg a.b$



jeremie.hamon@imag.fr

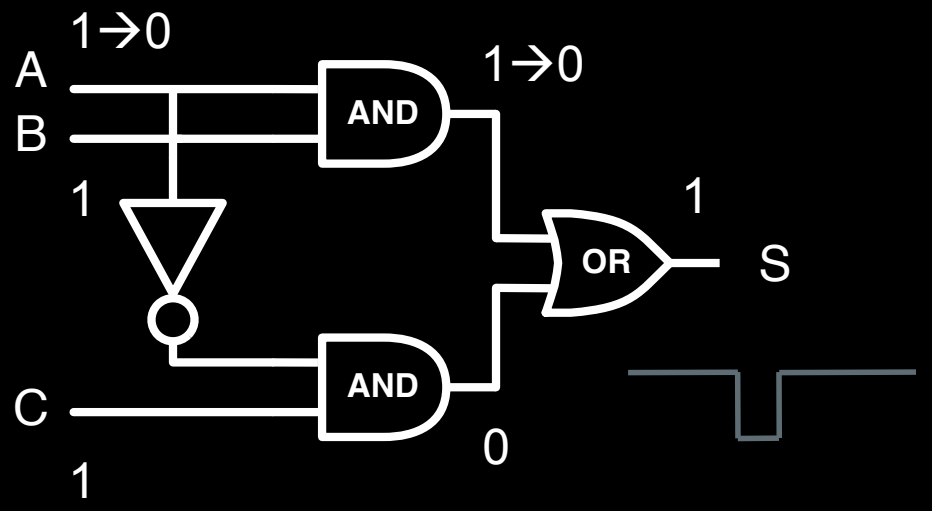
Conception sans aléa

Exemple : Single Input Change

- Tableau de Karnaugh

cb \ a	0	1
00	0	0
01	0	1
11	1	1
10	1	0

$s = a.c + \neg a.b$



jeremie.hamon@imag.fr

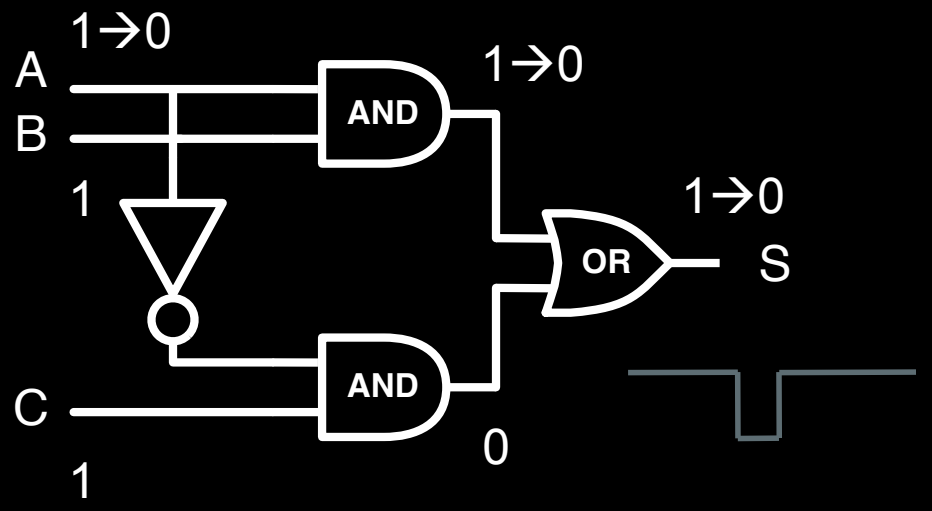
Conception sans aléa

Exemple : Single Input Change

- Tableau de Karnaugh

cb \ a	0	1
00	0	0
01	0	1
11	1	1
10	1	0

$s = a.c + \neg a.b$



jeremie.hamon@imag.fr

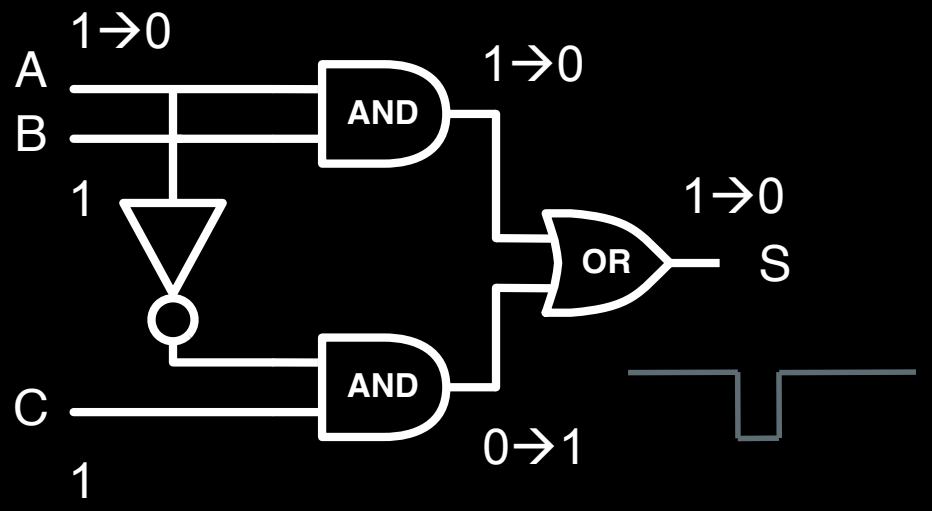
Conception sans aléa

Exemple : Single Input Change

- Tableau de Karnaugh

cb \ a	0	1
00	0	0
01	0	1
11	1	1
10	1	0

$s = a.c + \neg a.b$



jeremie.hamon@imag.fr

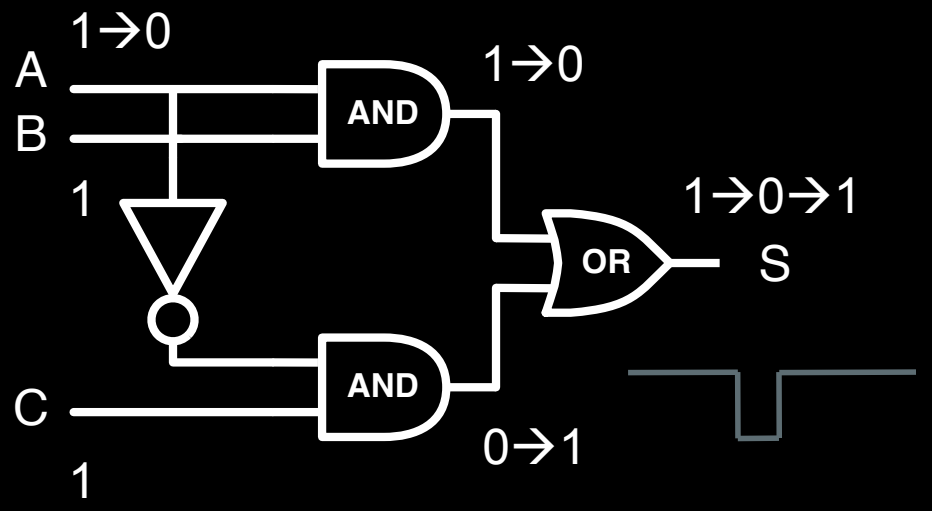
Conception sans aléa

Exemple : Single Input Change

- Tableau de Karnaugh

cb \ a	0	1
00	0	0
01	0	1
11	1	1
10	1	0

$s = a.c + \neg a.b$



jeremie.hamon@imag.fr

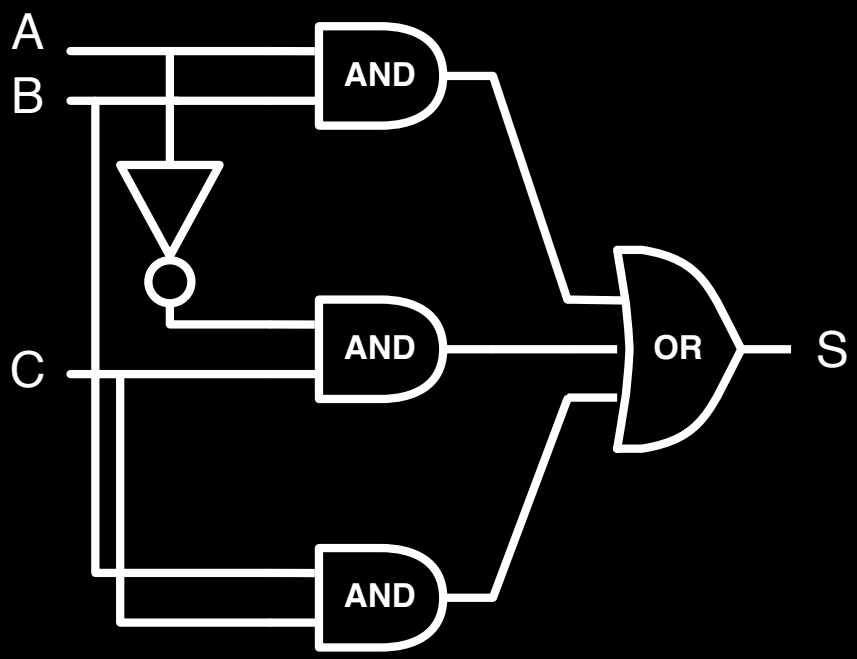
Conception sans aléa

Exemple : Single Input Change

- Tableau de Karnaugh

cb \ a	0	1
00	0	0
01	0	1
11	1	1
10	1	0

$s = a.c + \neg a.b + b.c$



jeremie.hamon@imag.fr

Conception sans aléa

Conclusion

- Aléas SIC : Implémentation ET-OU (somme de produits):
  - Transitions 0 → 0 : aucun aléa
  - Transitions 0 → 1 : aucun aléa
  - Transitions 1 → 0 : aucun aléa
  - Transitions 1 → 1 : suppression des aléas statiques (couverture sur le tableau de Karnaugh)
- Aléas MIC : même procédure mais avec plus de cas possibles!
- Eviter les aléas ⇔ Insérer de la redondance sur le tableau de Karnaugh

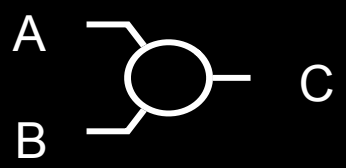
Deuxième partie

Introduction à la conception en logique asynchrone

- Conception sans aléas
- La porte de Muller
- Synthèse de la logique QDI
  - Logique DIMS
- Eléments de mémorisation
  - Le Half-Buffer
- Architecture des circuits asynchrones
  - Pipeline asynchrone linéaire
  - Pipeline asynchrone non-linéaire
  - Deadlocks

La porte de Muller – « C element »

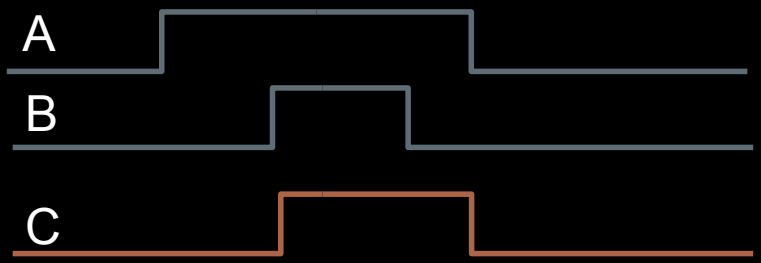
- Fonction : Rendez-vous de plusieurs signaux
- Symbole :



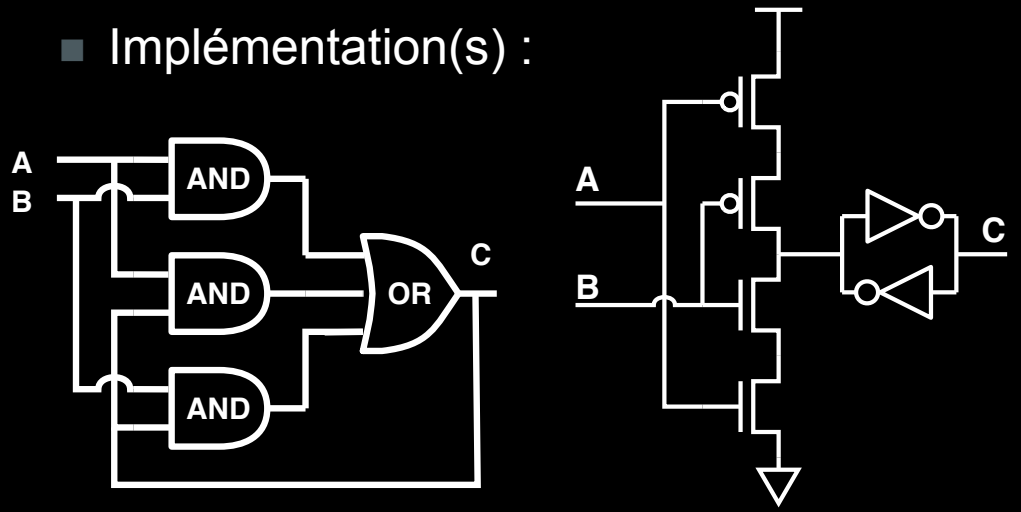
- Equation : $S = A.B + S(A+B)$

A	B	S
0	0	0
0	1	S^{-1}
1	0	S^{-1}
1	1	1

- Chronogrammes :



- Implémentation(s) :



jeremie.hamon@imag.fr

Deuxième partie

Introduction à la conception en logique asynchrone

- Conception sans aléas
- La porte de Muller
- Synthèse de la logique QDI
  - Logique DIMS
- Eléments de mémorisation
  - Le Half-Buffer
- Architecture des circuits asynchrones
  - Pipeline asynchrone linéaire
  - Pipeline asynchrone non-linéaire
  - Deadlocks

Synthèse de la logique QDI

Logique DIMS

- « Delay Insensitive Minterms Synthesis »
- Logique double rails
- Exemple : Porte XOR

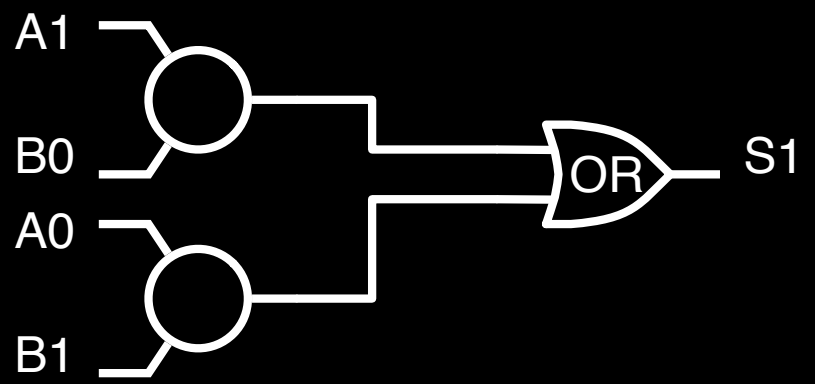
$$S = A \text{ xor } B = A.\neg B + \neg A.B$$

jeremie.hamon@imag.fr

Synthèse de la logique QDI

Logique DIMS

- « Delay Insensitive Minterms Synthesis »
- Logique double rails
- Exemple : Porte XOR



S = A xor B = A.¬B + ¬A.B

Rail 1

S1 = A1.B0 + A0.B1

jeremie.hamon@imag.fr

Synthèse de la logique QDI

Logique DIMS

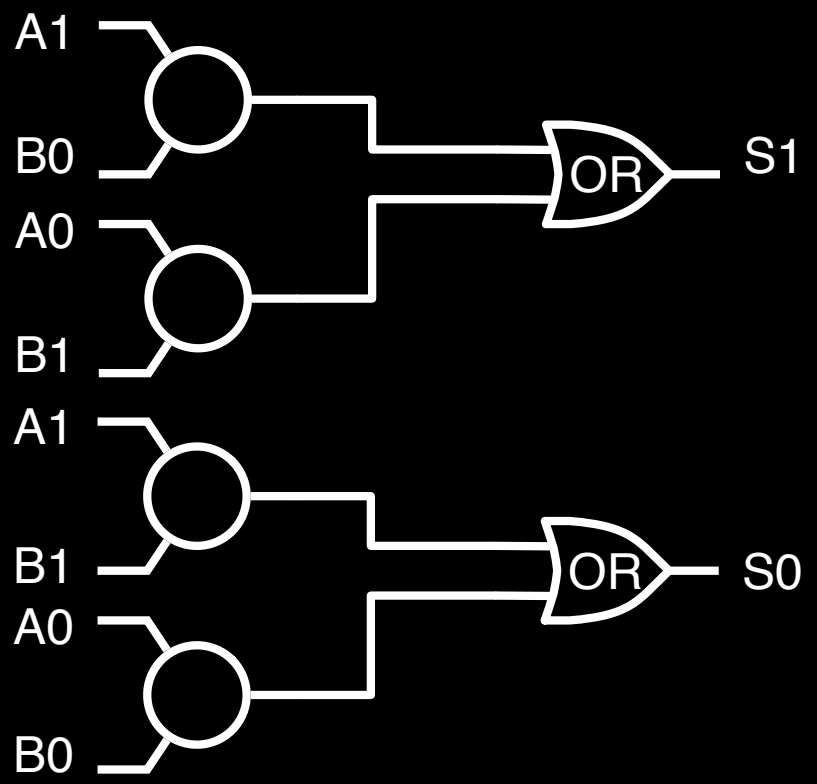
- « Delay Insensitive Minterms Synthesis »
- Logique double rails
- Exemple : Porte XOR

S = A xor B = A.¬B + ¬A.B

Rail 0

S1 = A1.B0 + A0.B1

S0 = A0.B1 + A1.B0

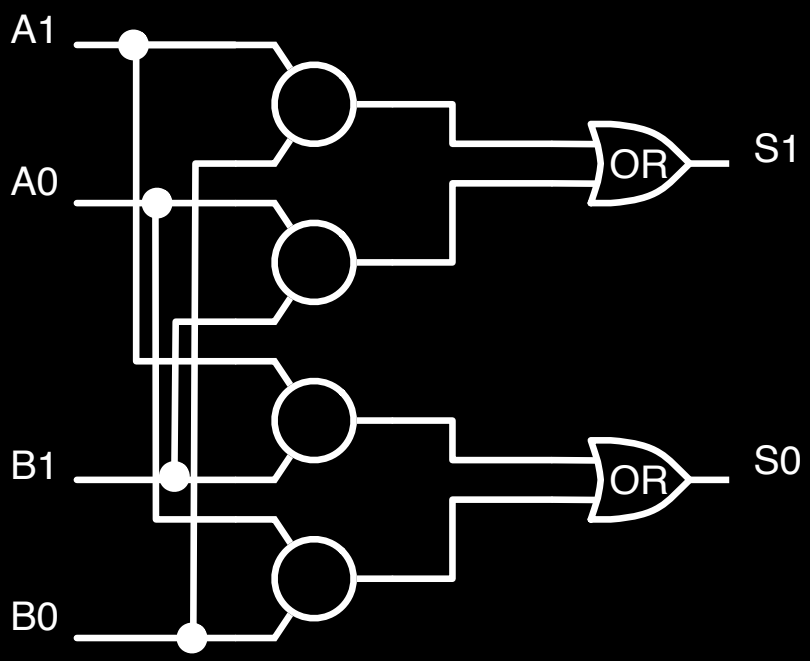


jeremie.hamon@imag.fr

Synthèse de la logique QDI

Logique DIMS - Conclusion

- Implémentation directe ☺
- Vitesse
- Surface du circuit ☹
  - Nb Mullers = $2^{\text{nb inputs}}$
- Techniques d'optimisation
  - Réduction (MDD)
  - Factorisation (MDD)
  - Portes Complexes (Tech Mapping)



Porte XOR 2 entrées QDI - DIMS

jeremie.hamon@imag.fr

Deuxième partie

Introduction à la conception en logique asynchrone

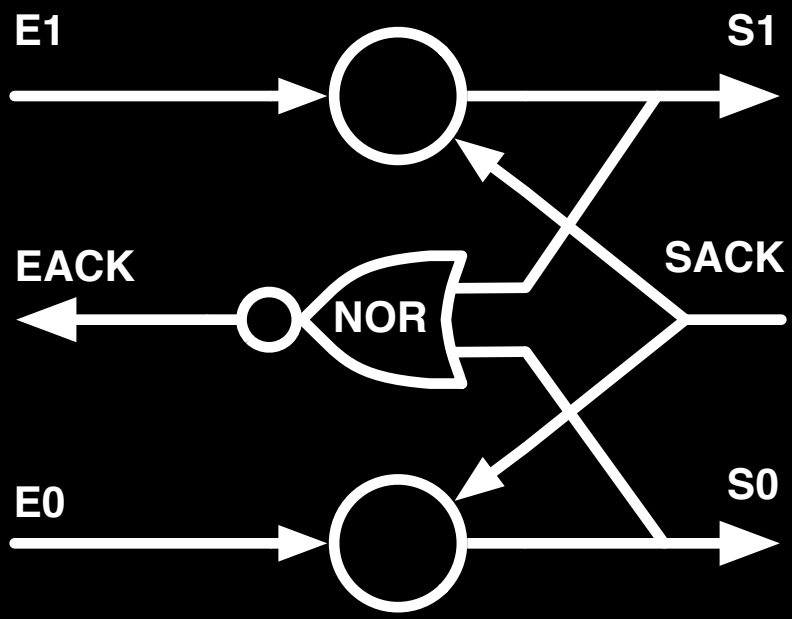
- Conception sans aléas
- La porte de Muller
- Synthèse de la logique QDI
  - Logique DIMS
- Eléments de mémorisation
  - Le Half-Buffer
- Architecture des circuits asynchrones
  - Pipeline asynchrone linéaire
  - Pipeline asynchrone non-linéaire
  - Deadlocks

jeremie.hamon@imag.fr

Eléments de mémorisation

Le « Half-Buffer »

- Mémorisation d'une donnée
  - Donnée valide (01 ou 10)
  - Donnée invalide (00)
- Implémentation du protocole de communication (4 phases) :
  - Protocole WCHB
  - Génération de l'acquittement
- « Half-Buffer »
  - Mémorisation de la moitié d'une donnée complète :
  - «Jeton valide & Jeton invalide»

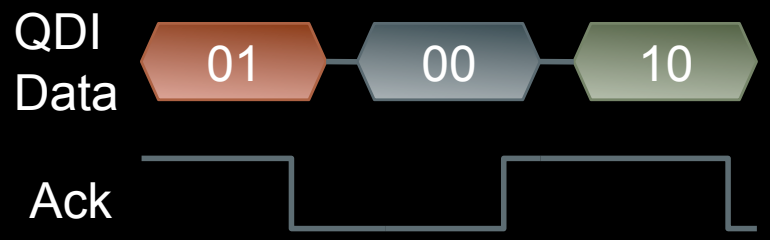


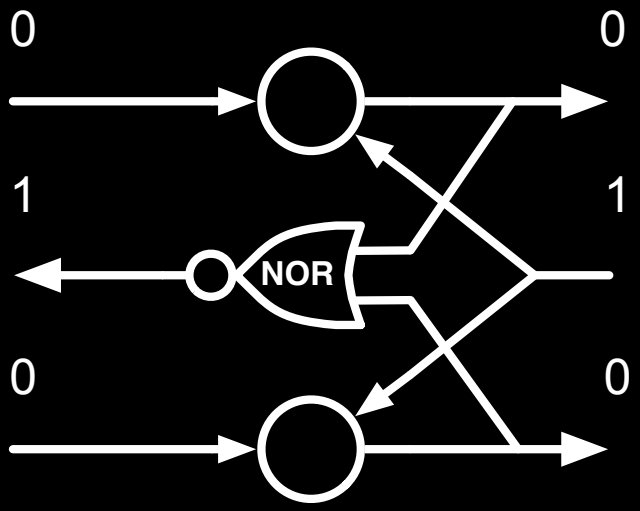
jeremie.hamon@imag.fr

Eléments de mémorisation

Le « Half-Buffer »

- Mémorisation d'une donnée
  - Donnée valide (01 ou 10)
  - Donnée invalide (00)
- Implémentation du protocole de communication (4 phases) :
  - Protocole WCHB
  - Génération de l'acquittement
- « Half-Buffer »
  - Mémorisation de la moitié d'une donnée complète :
  - «Jeton valide & Jeton invalide»



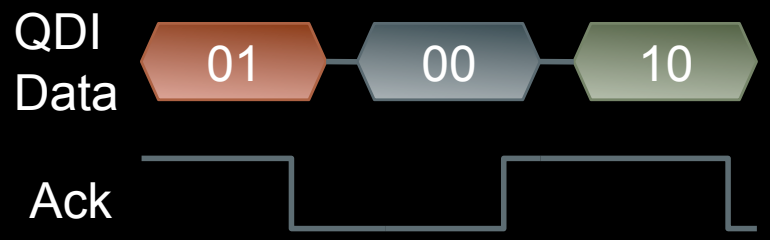


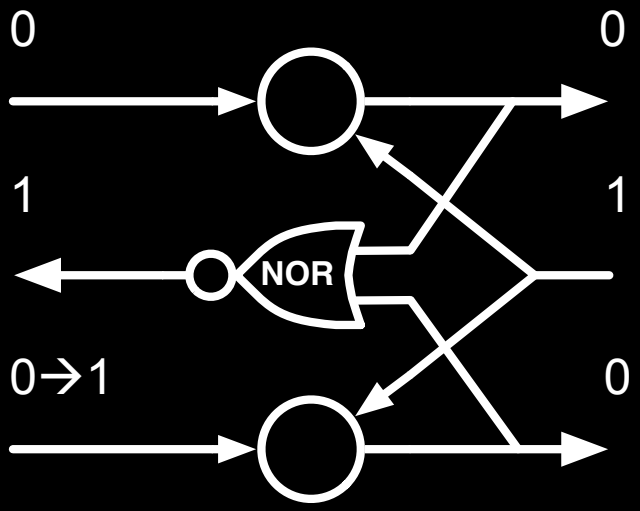
jeremie.hamon@imag.fr

Eléments de mémorisation

Le « Half-Buffer »

- Mémorisation d'une donnée
  - Donnée valide (01 ou 10)
  - Donnée invalide (00)
- Implémentation du protocole de communication (4 phases) :
  - Protocole WCHB
  - Génération de l'acquittement
- « Half-Buffer »
  - Mémorisation de la moitié d'une donnée complète :
  - «Jeton valide & Jeton invalide»



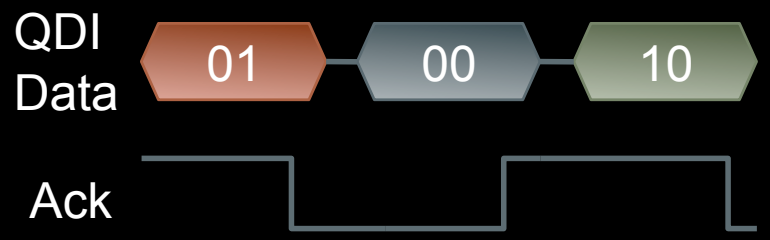


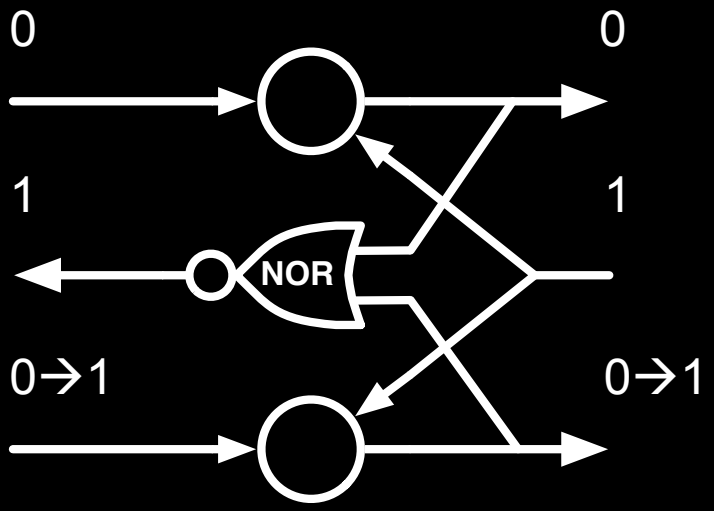
jeremie.hamon@imag.fr

Eléments de mémorisation

Le « Half-Buffer »

- Mémorisation d'une donnée
  - Donnée valide (01 ou 10)
  - Donnée invalide (00)
- Implémentation du protocole de communication (4 phases) :
  - Protocole WCHB
  - Génération de l'acquittement
- « Half-Buffer »
  - Mémorisation de la moitié d'une donnée complète :
  - «Jeton valide & Jeton invalide»



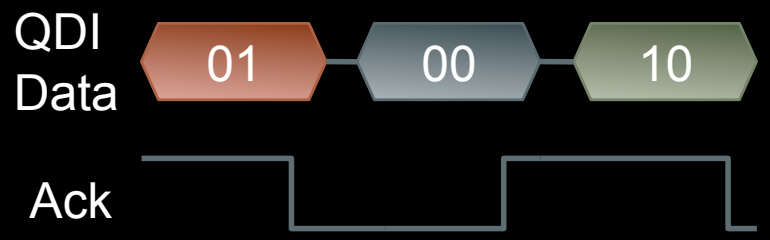


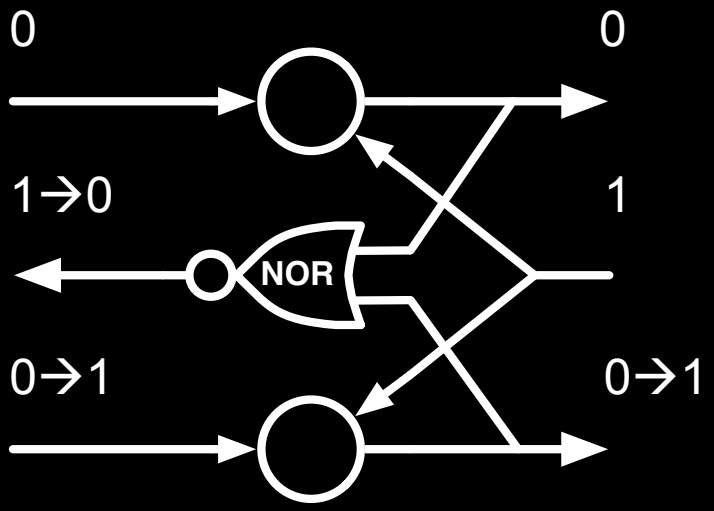
jeremie.hamon@imag.fr

Eléments de mémorisation

Le « Half-Buffer »

- Mémorisation d'une donnée
  - Donnée valide (01 ou 10)
  - Donnée invalide (00)
- Implémentation du protocole de communication (4 phases) :
  - Protocole WCHB
  - Génération de l'acquittement
- « Half-Buffer »
  - Mémorisation de la moitié d'une donnée complète :
  - «Jeton valide & Jeton invalide»



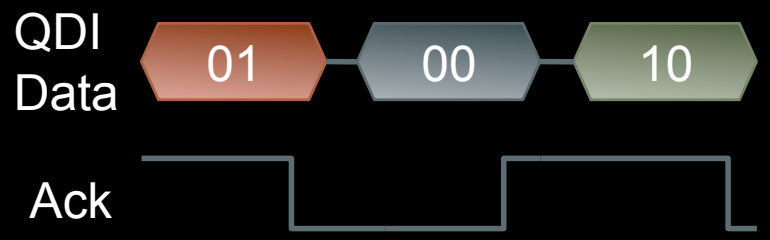


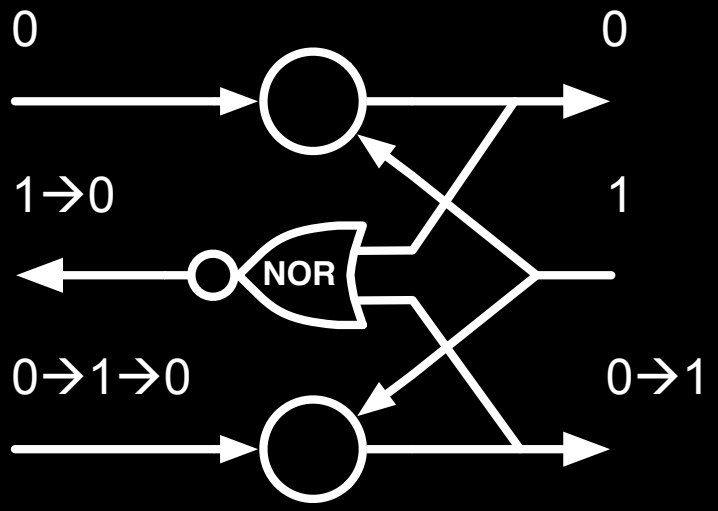
jeremie.hamon@imag.fr

Eléments de mémorisation

Le « Half-Buffer »

- Mémorisation d'une donnée
  - Donnée valide (01 ou 10)
  - Donnée invalide (00)
- Implémentation du protocole de communication (4 phases) :
  - Protocole WCHB
  - Génération de l'acquittement
- « Half-Buffer »
  - Mémorisation de la moitié d'une donnée complète :
  - «Jeton valide & Jeton invalide»



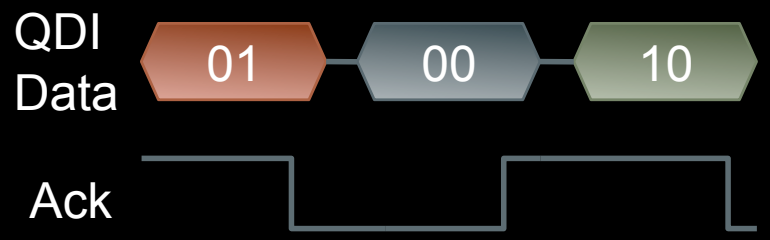


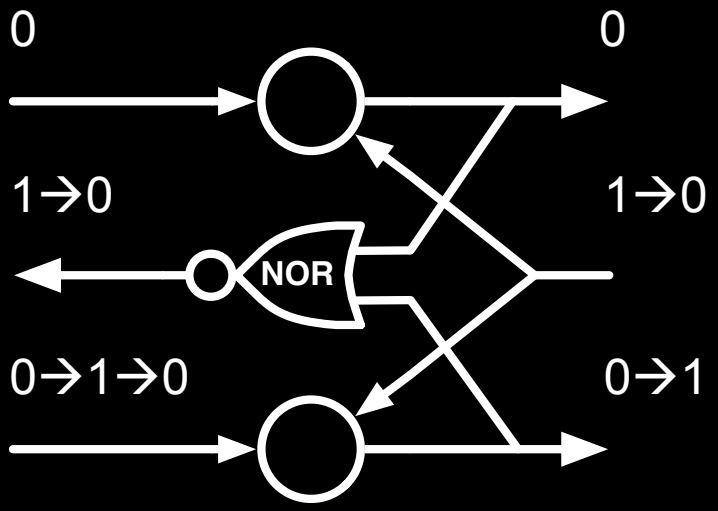
jeremie.hamon@imag.fr

Eléments de mémorisation

Le « Half-Buffer »

- Mémorisation d'une donnée
  - Donnée valide (01 ou 10)
  - Donnée invalide (00)
- Implémentation du protocole de communication (4 phases) :
  - Protocole WCHB
  - Génération de l'acquittement
- « Half-Buffer »
  - Mémorisation de la moitié d'une donnée complète :
  - «Jeton valide & Jeton invalide»



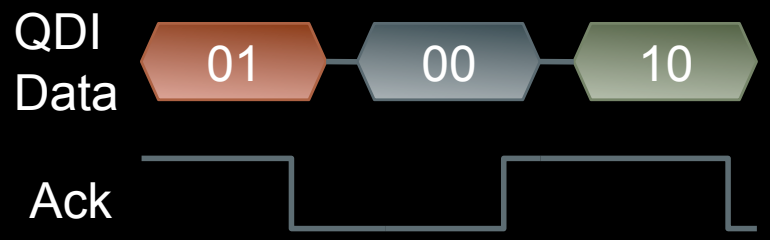


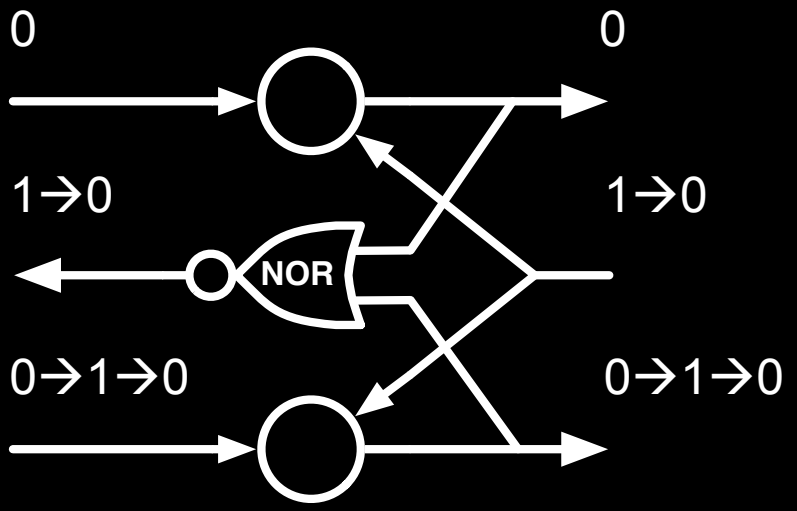
jeremie.hamon@imag.fr

Eléments de mémorisation

Le « Half-Buffer »

- Mémorisation d'une donnée
  - Donnée valide (01 ou 10)
  - Donnée invalide (00)
- Implémentation du protocole de communication (4 phases) :
  - Protocole WCHB
  - Génération de l'acquittement
- « Half-Buffer »
  - Mémorisation de la moitié d'une donnée complète :
  - «Jeton valide & Jeton invalide»



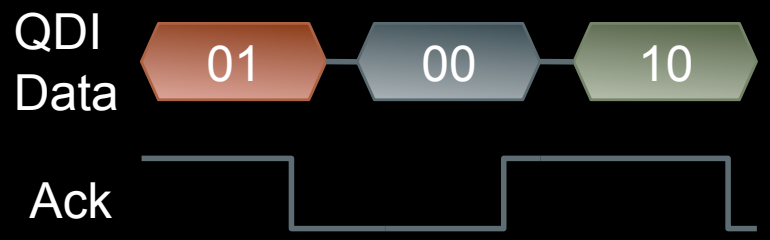


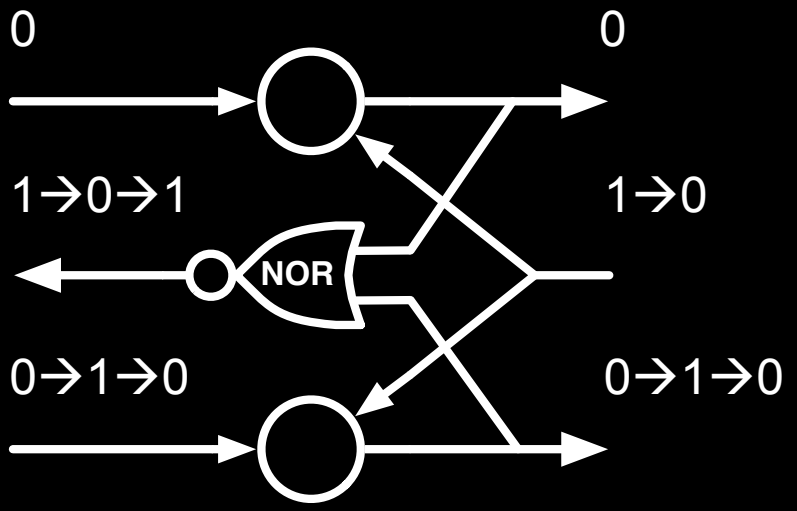
jeremie.hamon@imag.fr

Eléments de mémorisation

Le « Half-Buffer »

- Mémorisation d'une donnée
  - Donnée valide (01 ou 10)
  - Donnée invalide (00)
- Implémentation du protocole de communication (4 phases) :
  - Protocole WCHB
  - Génération de l'acquittement
- « Half-Buffer »
  - Mémorisation de la moitié d'une donnée complète :
  - «Jeton valide & Jeton invalide»





jeremie.hamon@imag.fr

Deuxième partie

Introduction à la conception en logique asynchrone

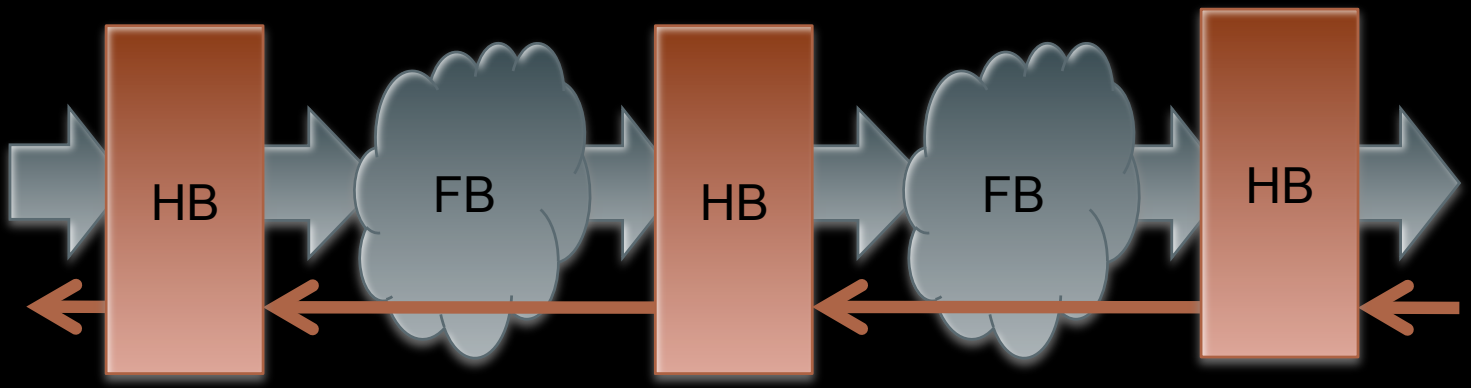
- Conception sans aléas
- La porte de Muller
- Synthèse de la logique QDI
  - Logique DIMS
- Eléments de mémorisation
  - Le Half-Buffer
- Architecture des circuits asynchrones
  - Pipeline asynchrone linéaire
  - Pipeline asynchrone non-linéaire
  - Deadlocks

jeremie.hamon@imag.fr

Architecture des circuits QDI

Pipeline asynchrone linéaire

- Blocs fonctionnels (DIMS) + Eléments de mémorisation (HB)
- « Token Game » :
  - Représentation du flot de données par des jetons et des bulles
  - Jeton : étage mémorisant une donnée (valide ou invalide)
  - Bulle : étage vide prêt à mémoriser une nouvelle donnée

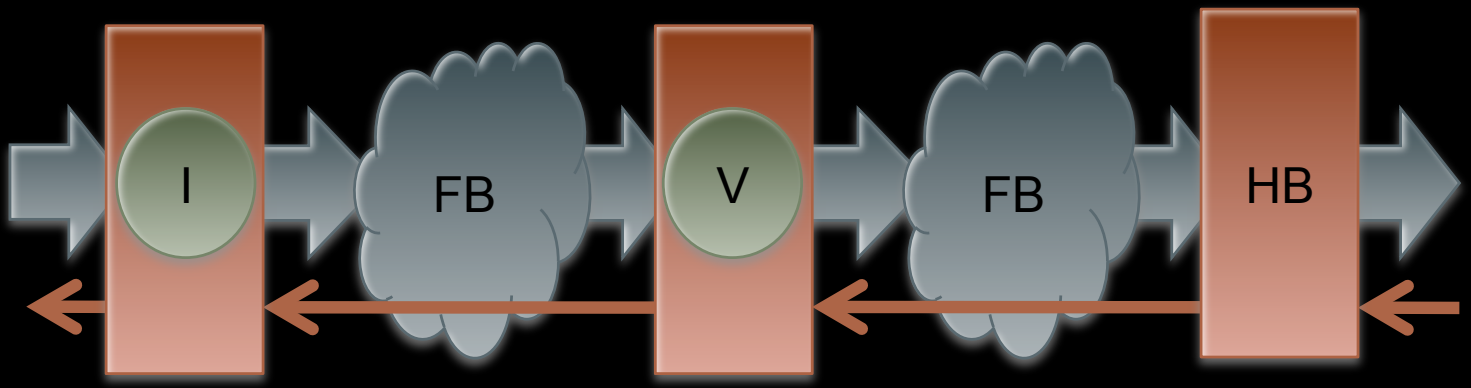


jeremie.hamon@imag.fr

Architecture des circuits QDI

Pipeline asynchrone linéaire

- Blocs fonctionnels (DIMS) + Eléments de mémorisation (HB)
- « Token Game » :
  - Représentation du flot de données par des jetons et des bulles
  - Jeton : étage mémorisant une donnée (valide ou invalide)
  - Bulle : étage vide prêt à mémoriser une nouvelle donnée

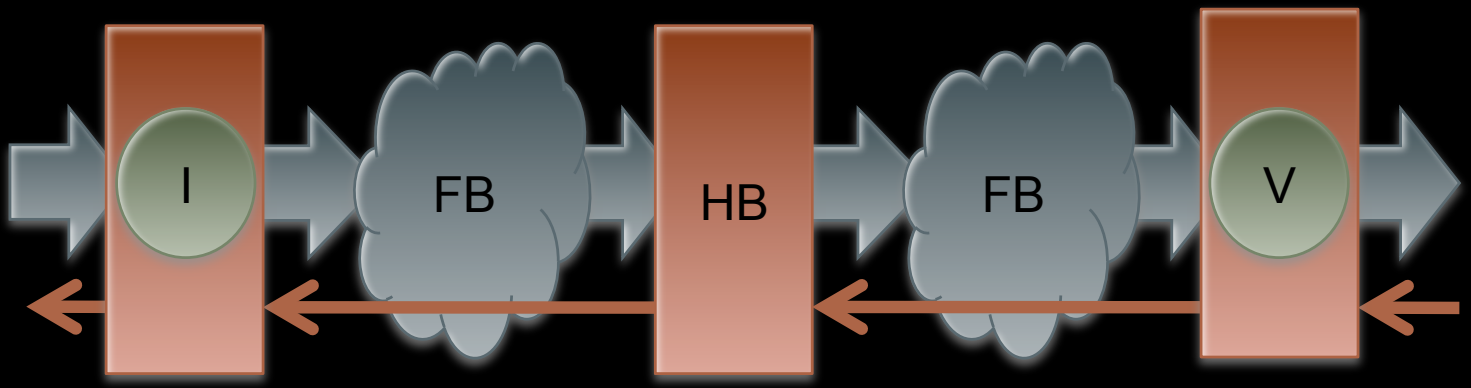


jeremie.hamon@imag.fr

Architecture des circuits QDI

Pipeline asynchrone linéaire

- Blocs fonctionnels (DIMS) + Eléments de mémorisation (HB)
- « Token Game » :
  - Représentation du flot de données par des jetons et des bulles
  - Jeton : étage mémorisant une donnée (valide ou invalide)
  - Bulle : étage vide prêt à mémoriser une nouvelle donnée

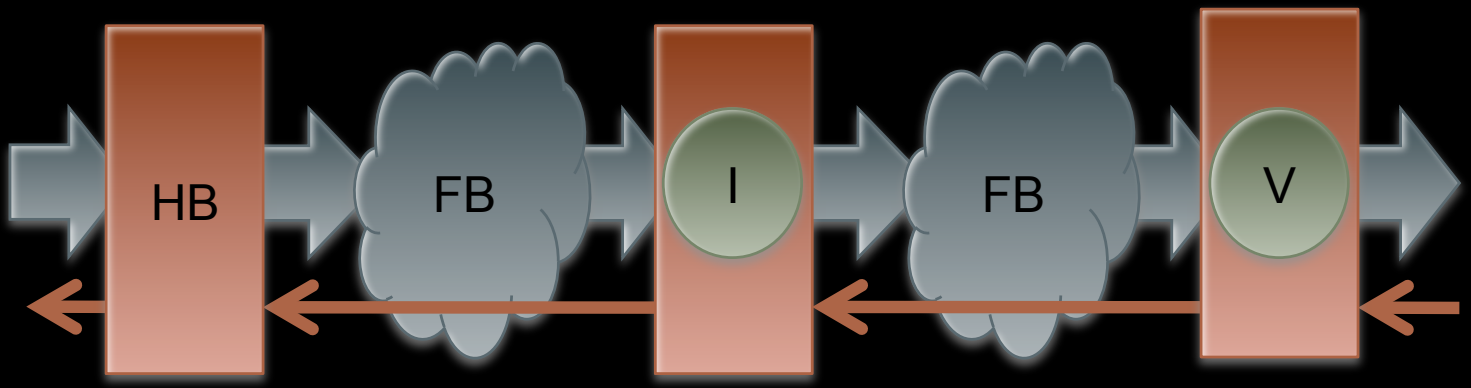


jeremie.hamon@imag.fr

Architecture des circuits QDI

Pipeline asynchrone linéaire

- Blocs fonctionnels (DIMS) + Eléments de mémorisation (HB)
- « Token Game » :
  - Représentation du flot de données par des jetons et des bulles
  - Jeton : étage mémorisant une donnée (valide ou invalide)
  - Bulle : étage vide prêt à mémoriser une nouvelle donnée

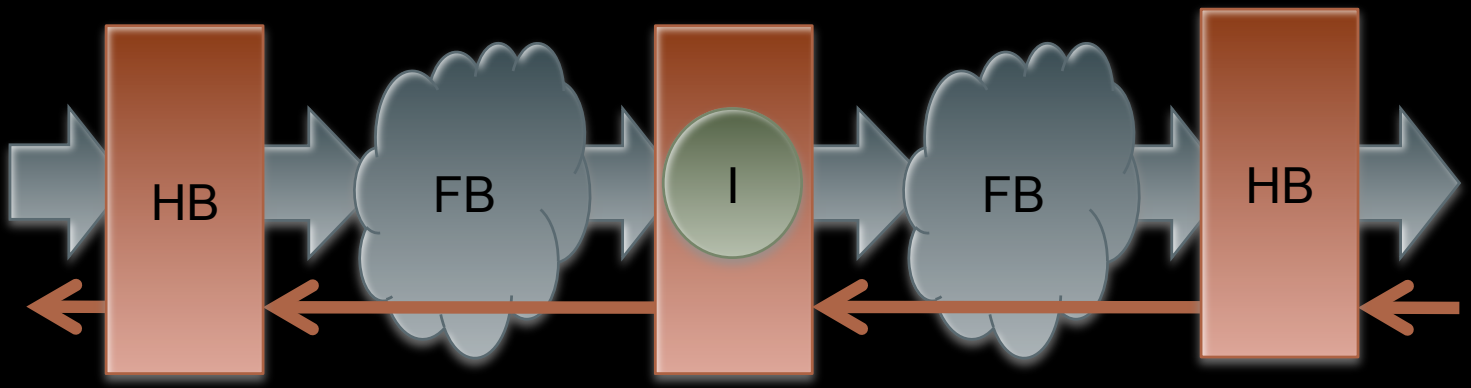


jeremie.hamon@imag.fr

Architecture des circuits QDI

Pipeline asynchrone linéaire

- Blocs fonctionnels (DIMS) + Eléments de mémorisation (HB)
- « Token Game » :
  - Représentation du flot de données par des jetons et des bulles
  - Jeton : étage mémorisant une donnée (valide ou invalide)
  - Bulle : étage vide prêt à mémoriser une nouvelle donnée

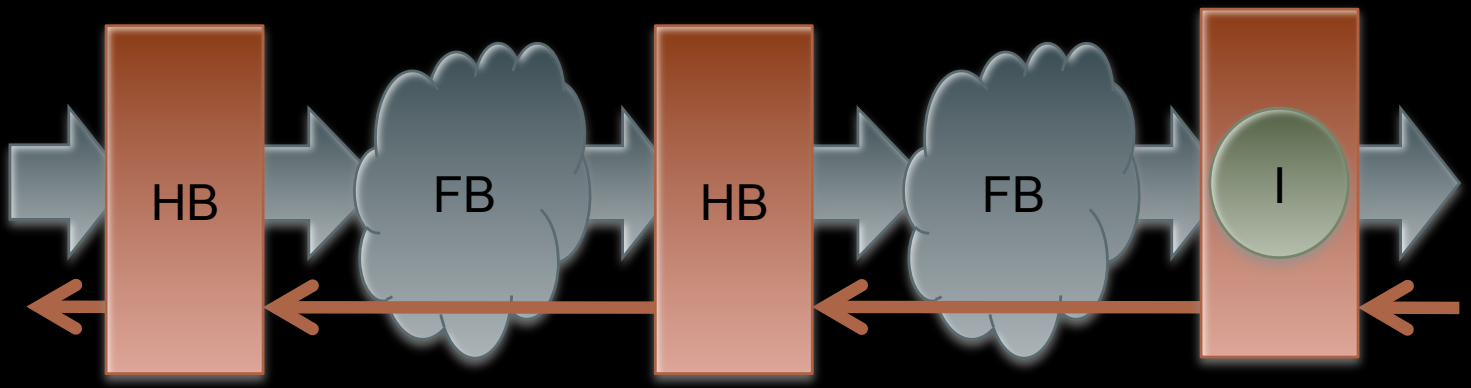


jeremie.hamon@imag.fr

Architecture des circuits QDI

Pipeline asynchrone linéaire

- Blocs fonctionnels (DIMS) + Eléments de mémorisation (HB)
- « Token Game » :
  - Représentation du flot de données par des jetons et des bulles
  - Jeton : étage mémorisant une donnée (valide ou invalide)
  - Bulle : étage vide prêt à mémoriser une nouvelle donnée

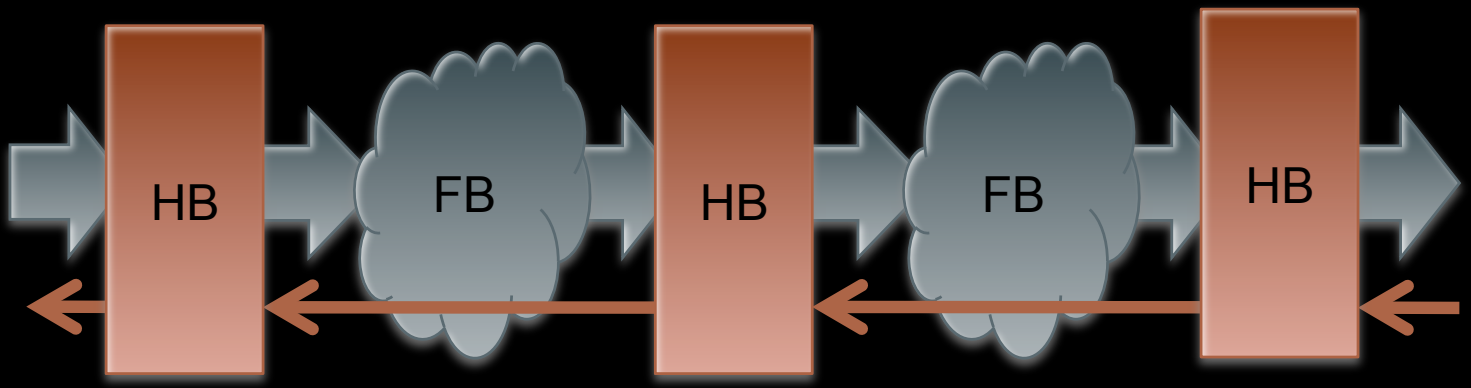


jeremie.hamon@imag.fr

Architecture des circuits QDI

Pipeline asynchrone linéaire

- Blocs fonctionnels (DIMS) + Eléments de mémorisation (HB)
- « Token Game » :
  - Représentation du flot de données par des jetons et des bulles
  - Jeton : étage mémorisant une donnée (valide ou invalide)
  - Bulle : étage vide prêt à mémoriser une nouvelle donnée

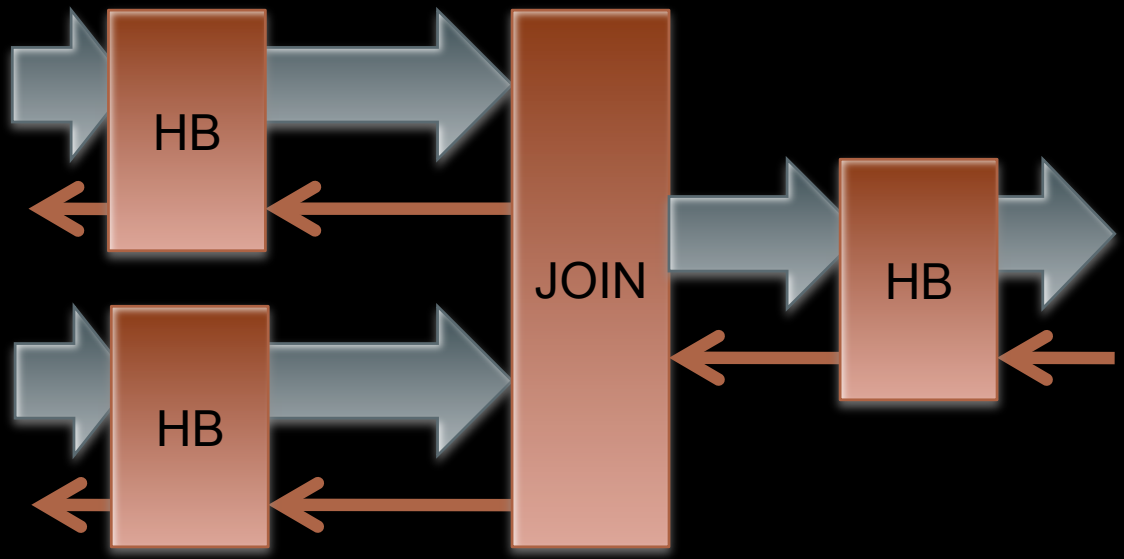


jeremie.hamon@imag.fr

Architecture des circuits QDI

Pipeline asynchrone non-linéaire

- Eléments non-lineaires :
  - JOIN : 2 jetons vers 1 jeton
  - FORK : 1 jeton vers 2 jetons

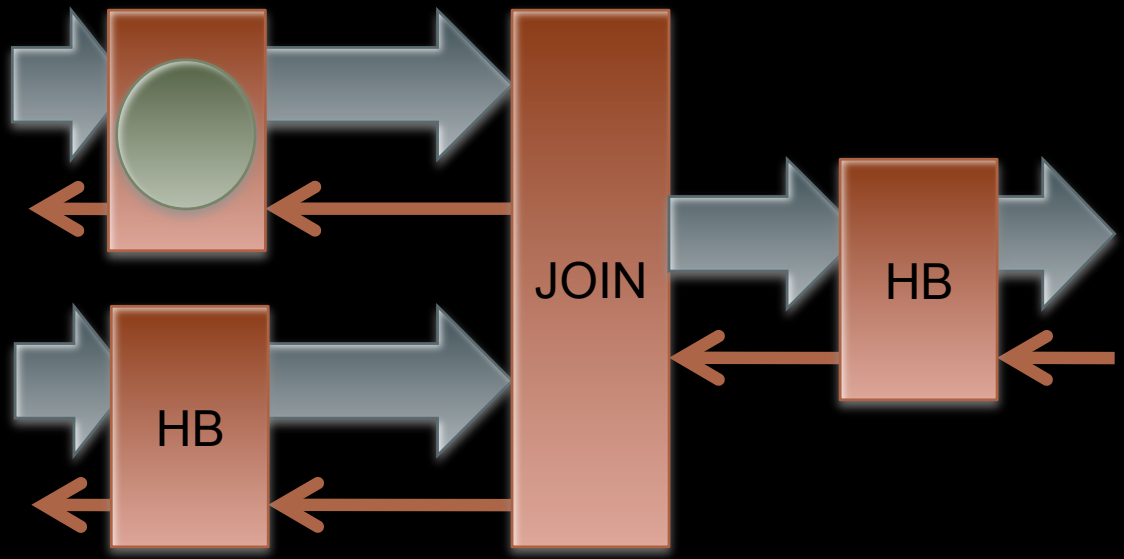


jeremie.hamon@imag.fr

Architecture des circuits QDI

Pipeline asynchrone non-linéaire

- Eléments non-lineaires :
  - JOIN : 2 jetons vers 1 jeton
  - FORK : 1 jeton vers 2 jetons

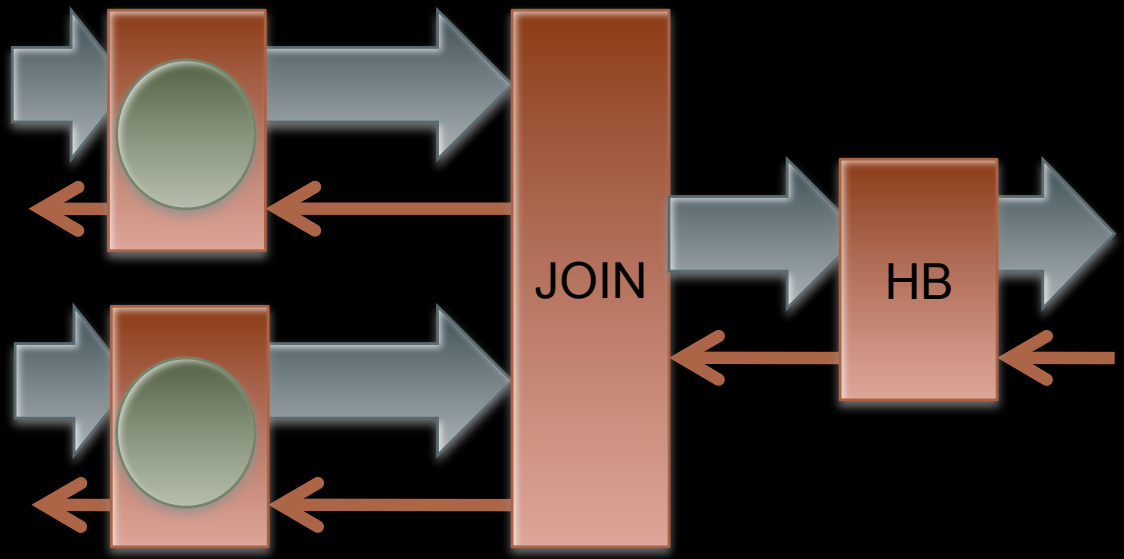


jeremie.hamon@imag.fr

Architecture des circuits QDI

Pipeline asynchrone non-linéaire

- Eléments non-lineaires :
  - JOIN : 2 jetons vers 1 jeton
  - FORK : 1 jeton vers 2 jetons

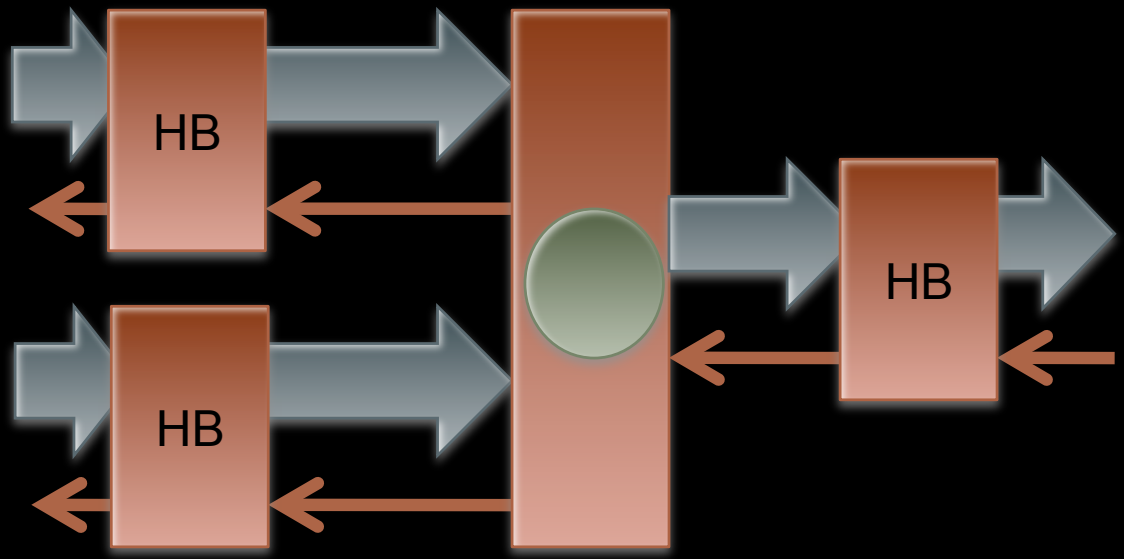


jeremie.hamon@imag.fr

Architecture des circuits QDI

Pipeline asynchrone non-linéaire

- Eléments non-lineaires :
  - JOIN : 2 jetons vers 1 jeton
  - FORK : 1 jeton vers 2 jetons

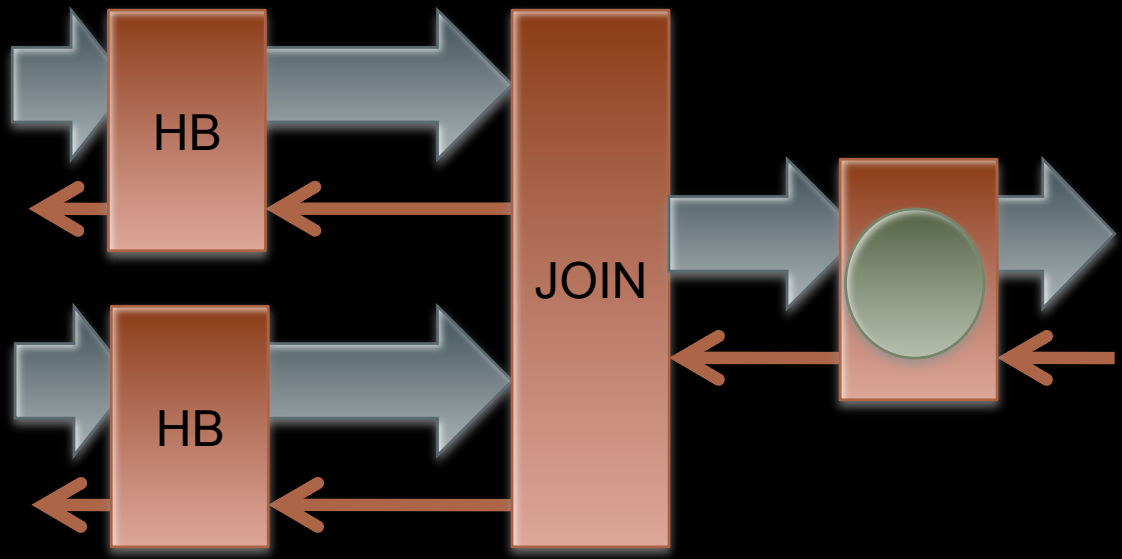


jeremie.hamon@imag.fr

Architecture des circuits QDI

Pipeline asynchrone non-linéaire

- Eléments non-lineaires :
  - JOIN : 2 jetons vers 1 jeton
  - FORK : 1 jeton vers 2 jetons

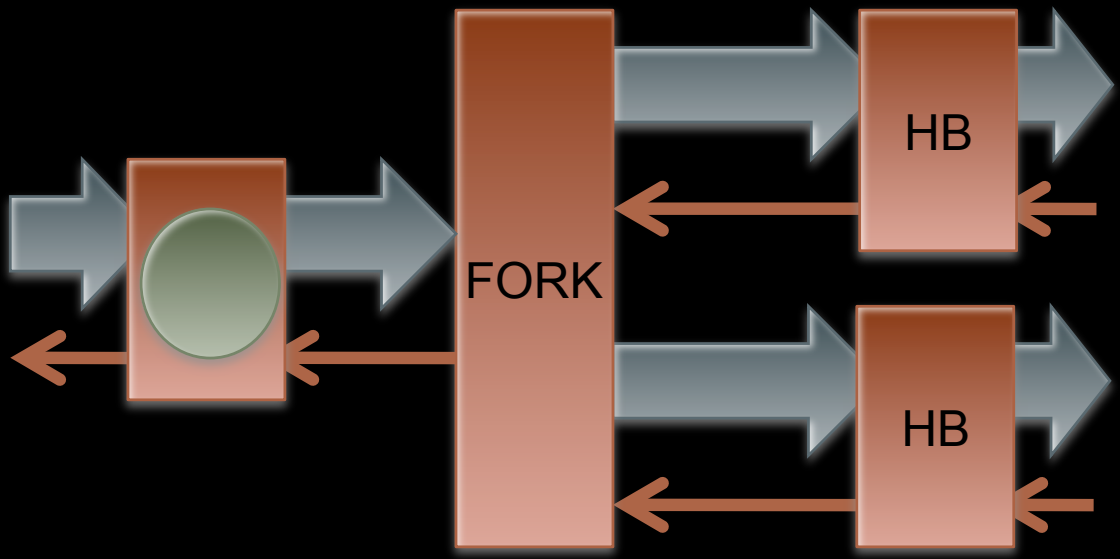


jeremie.hamon@imag.fr

Architecture des circuits QDI

Pipeline asynchrone non-linéaire

- Eléments non-lineaires :
  - JOIN : 2 jetons vers 1 jeton
  - FORK : 1 jeton vers 2 jetons

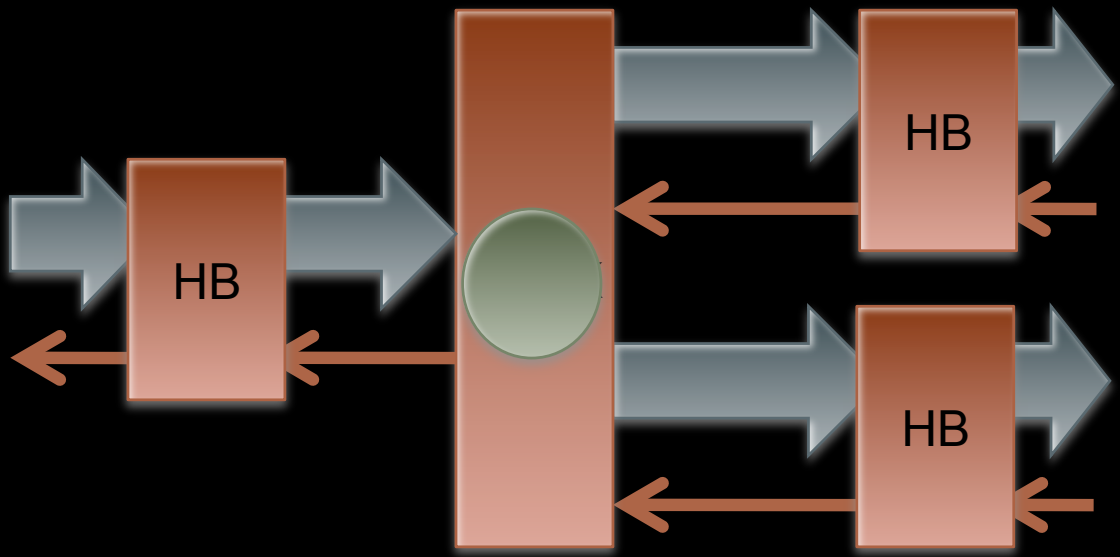


jeremie.hamon@imag.fr

Architecture des circuits QDI

Pipeline asynchrone non-linéaire

- Eléments non-lineaires :
  - JOIN : 2 jetons vers 1 jeton
  - FORK : 1 jeton vers 2 jetons

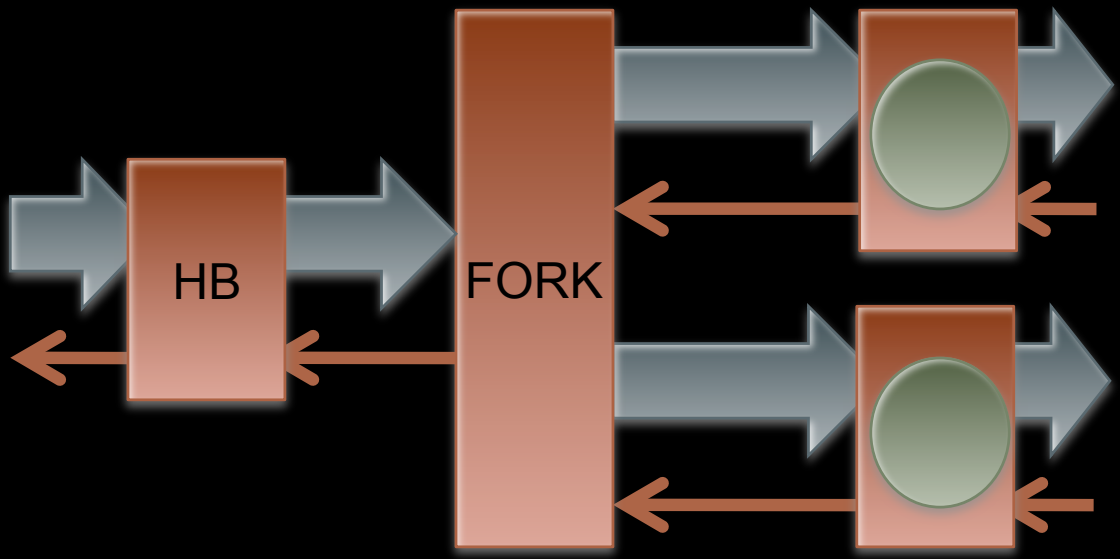


jeremie.hamon@imag.fr

Architecture des circuits QDI

Pipeline asynchrone non-linéaire

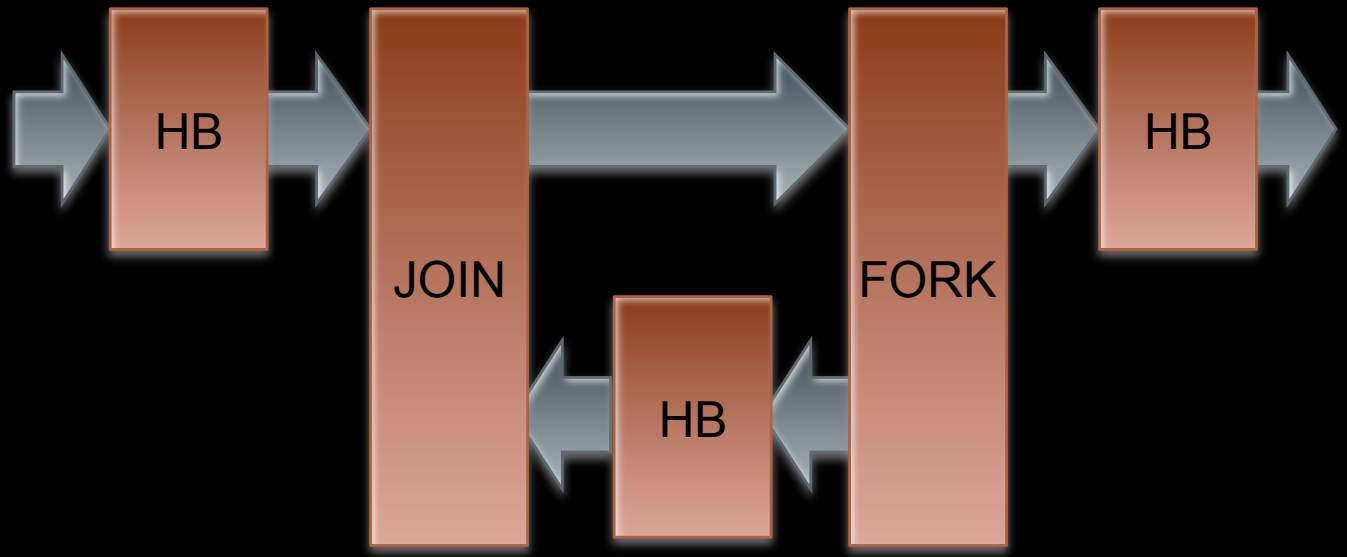
- Eléments non-lineaires :
  - JOIN : 2 jetons vers 1 jeton
  - FORK : 1 jeton vers 2 jetons



jeremie.hamon@imag.fr

Architecture des circuits QDI

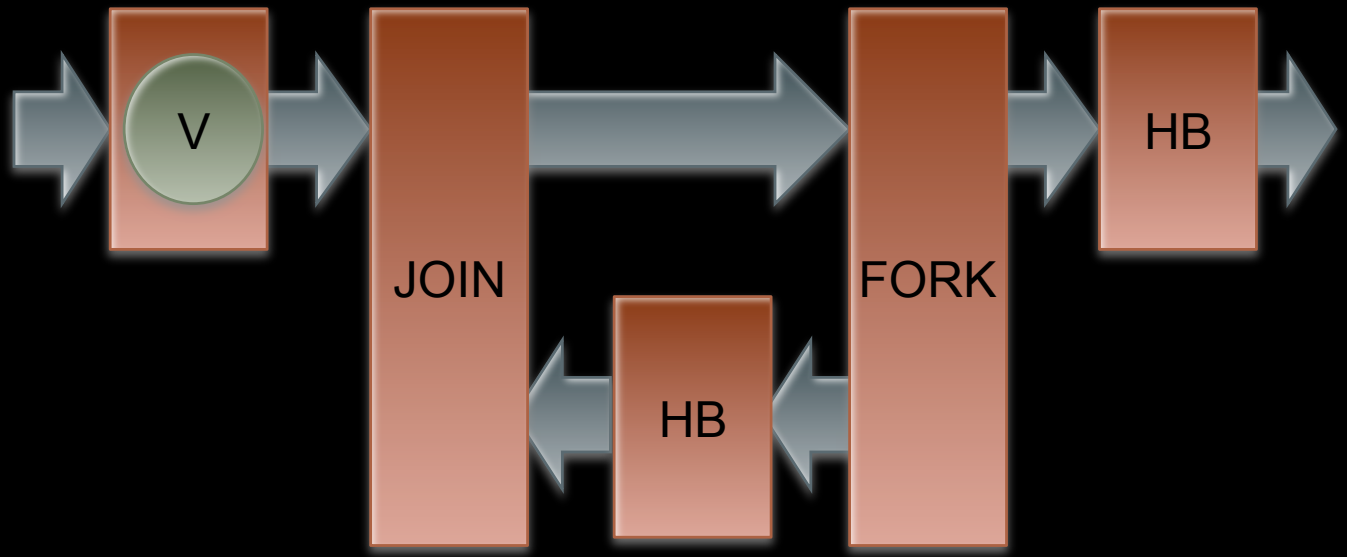
Deadlock



jeremie.hamon@imag.fr

Architecture des circuits QDI

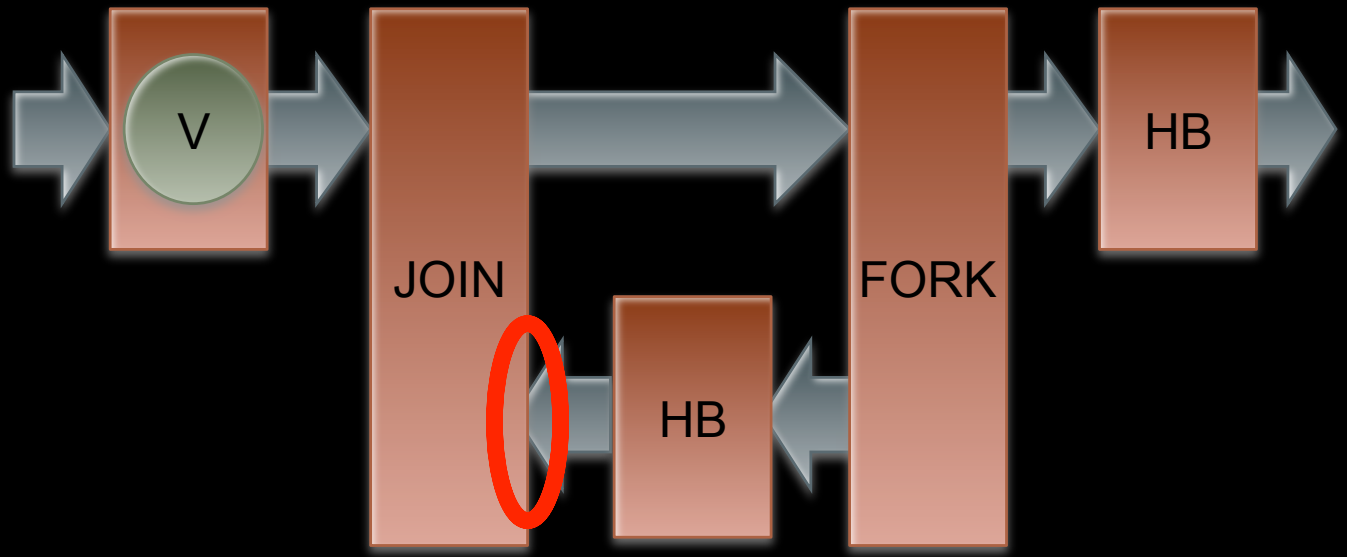
Deadlock



jeremie.hamon@imag.fr

Architecture des circuits QDI

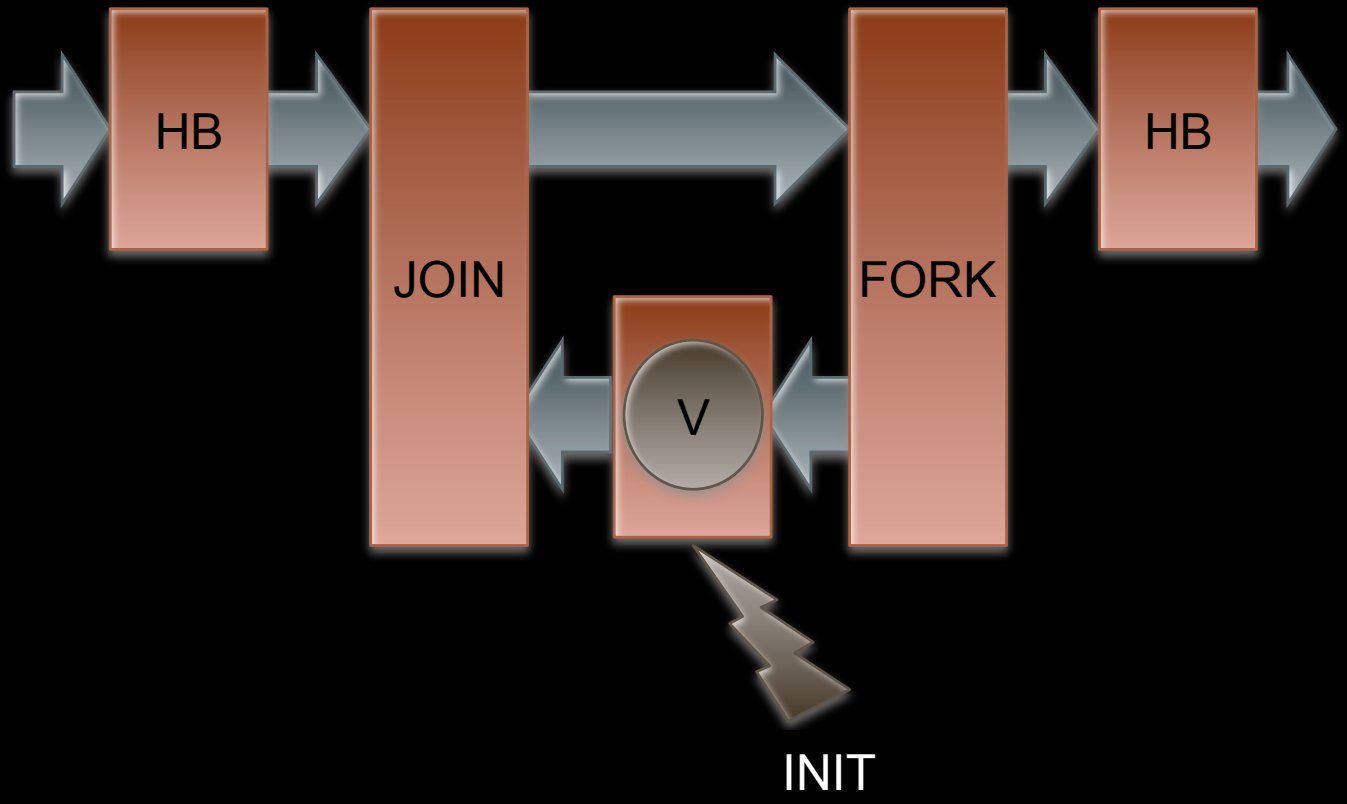
Deadlock



jeremie.hamon@imag.fr

Architecture des circuits QDI

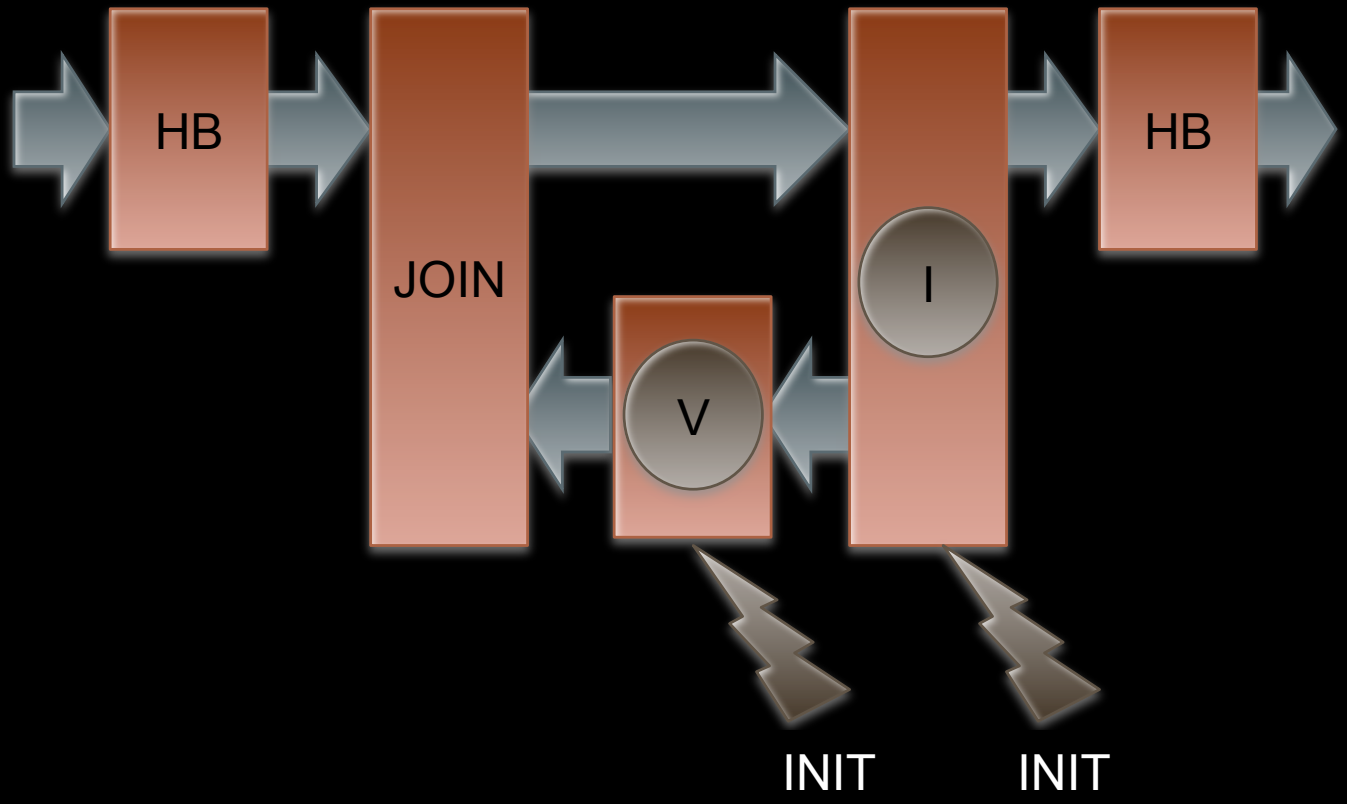
Deadlock



jeremie.hamon@imag.fr

Architecture des circuits QDI

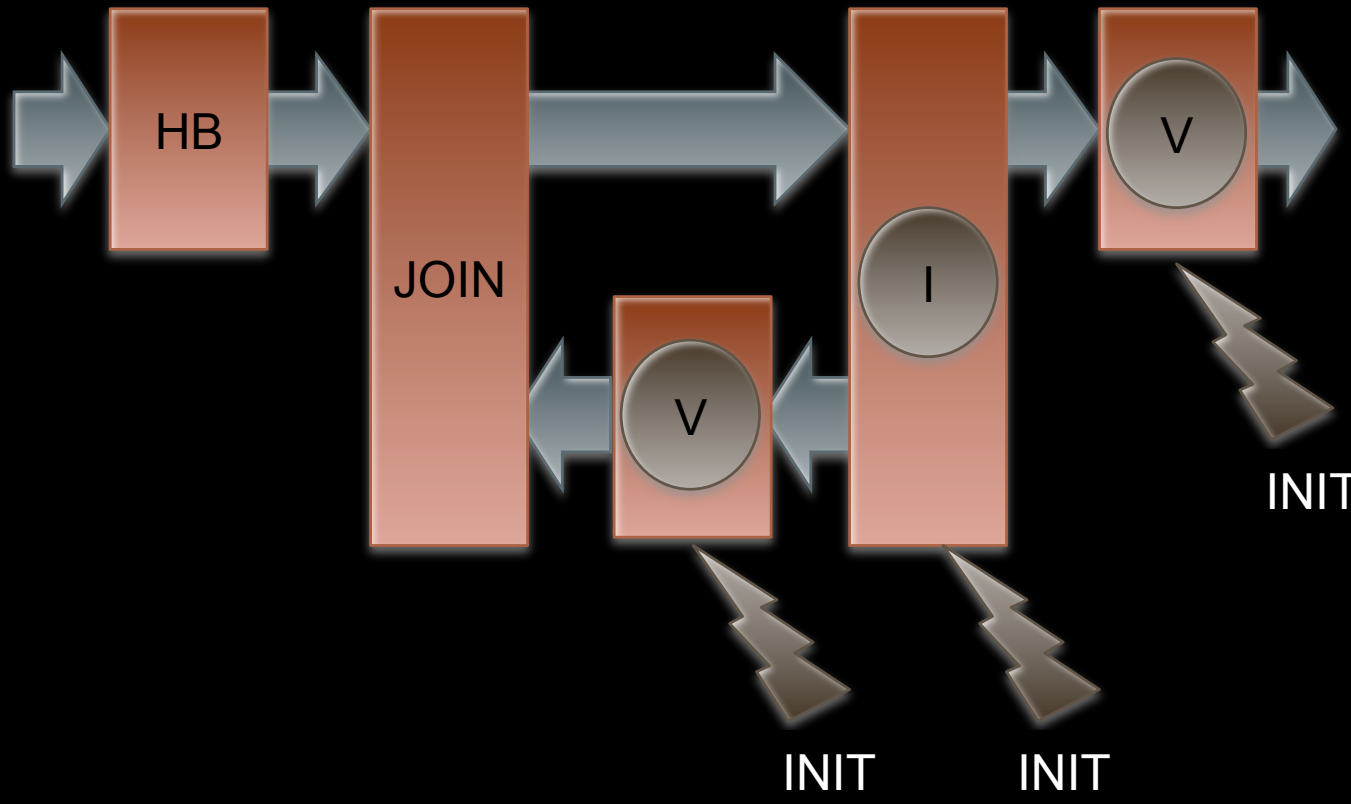
Deadlock



jeremie.hamon@imag.fr

Architecture des circuits QDI

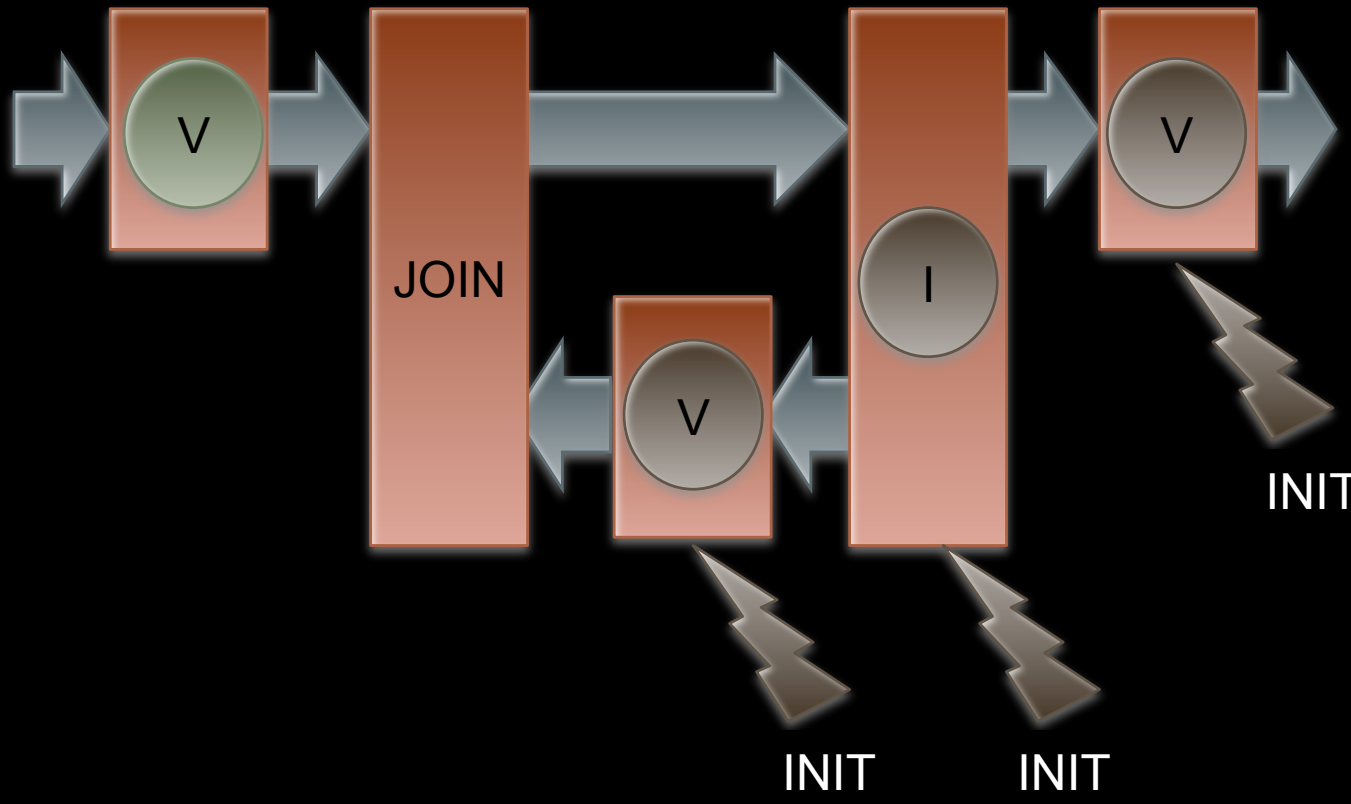
Deadlock



jeremie.hamon@imag.fr

Architecture des circuits QDI

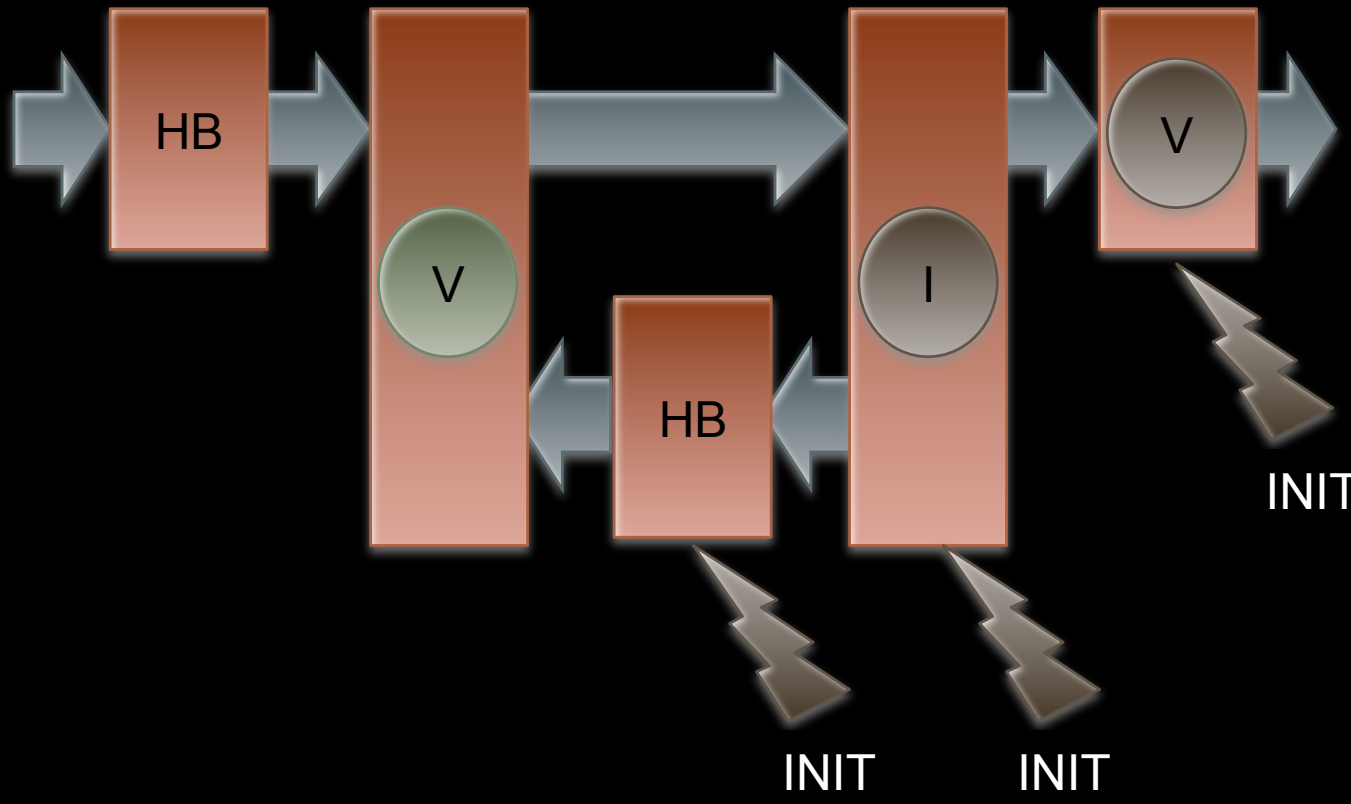
Deadlock



jeremie.hamon@imag.fr

Architecture des circuits QDI

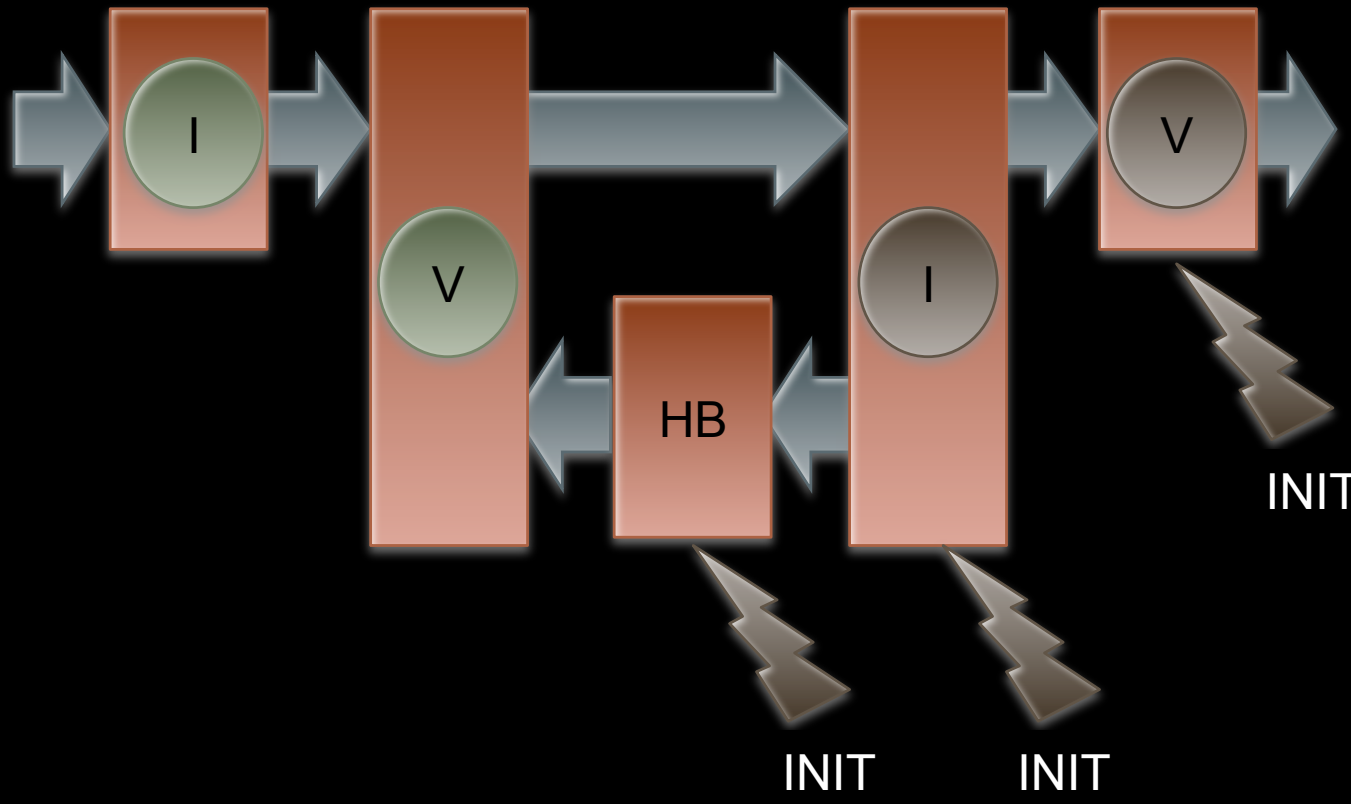
Deadlock



jeremie.hamon@imag.fr

Architecture des circuits QDI

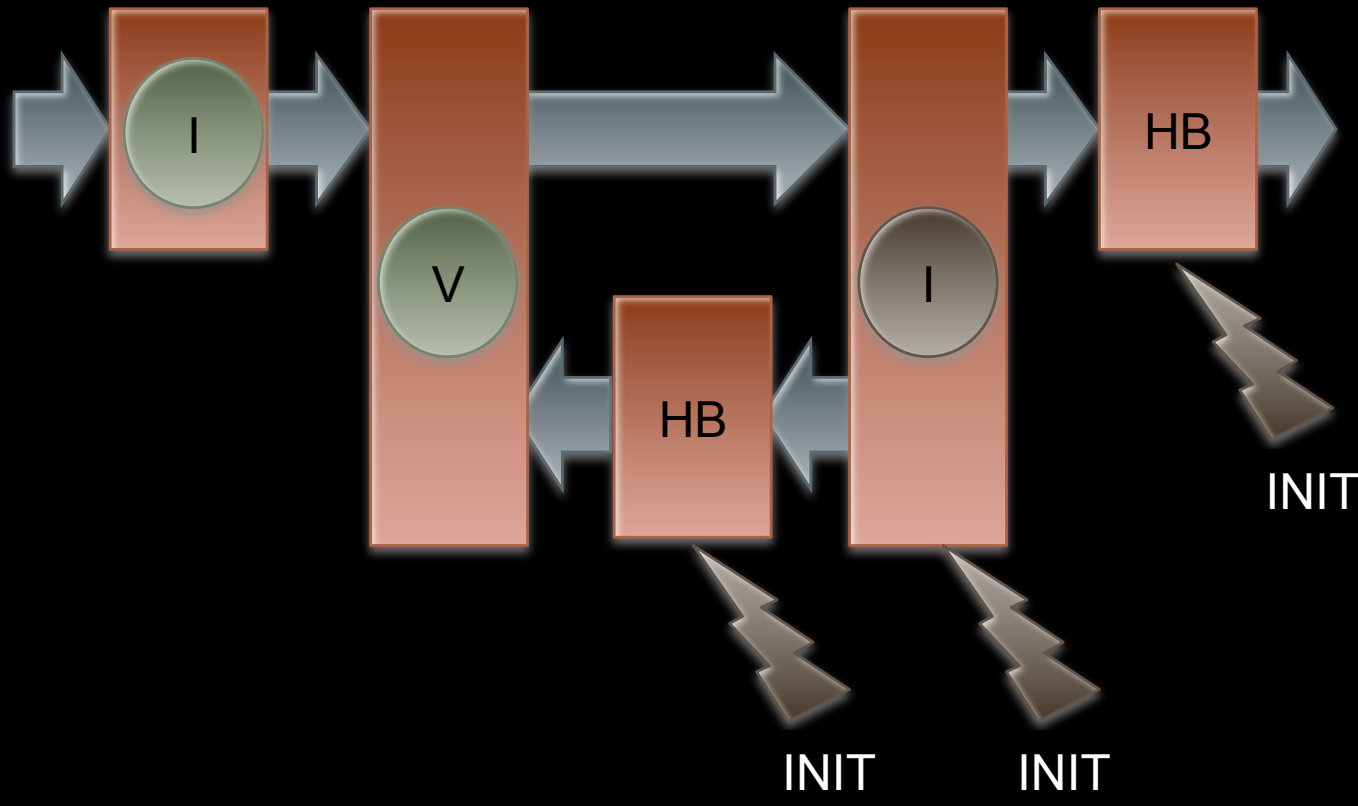
Deadlock



jeremie.hamon@imag.fr

Architecture des circuits QDI

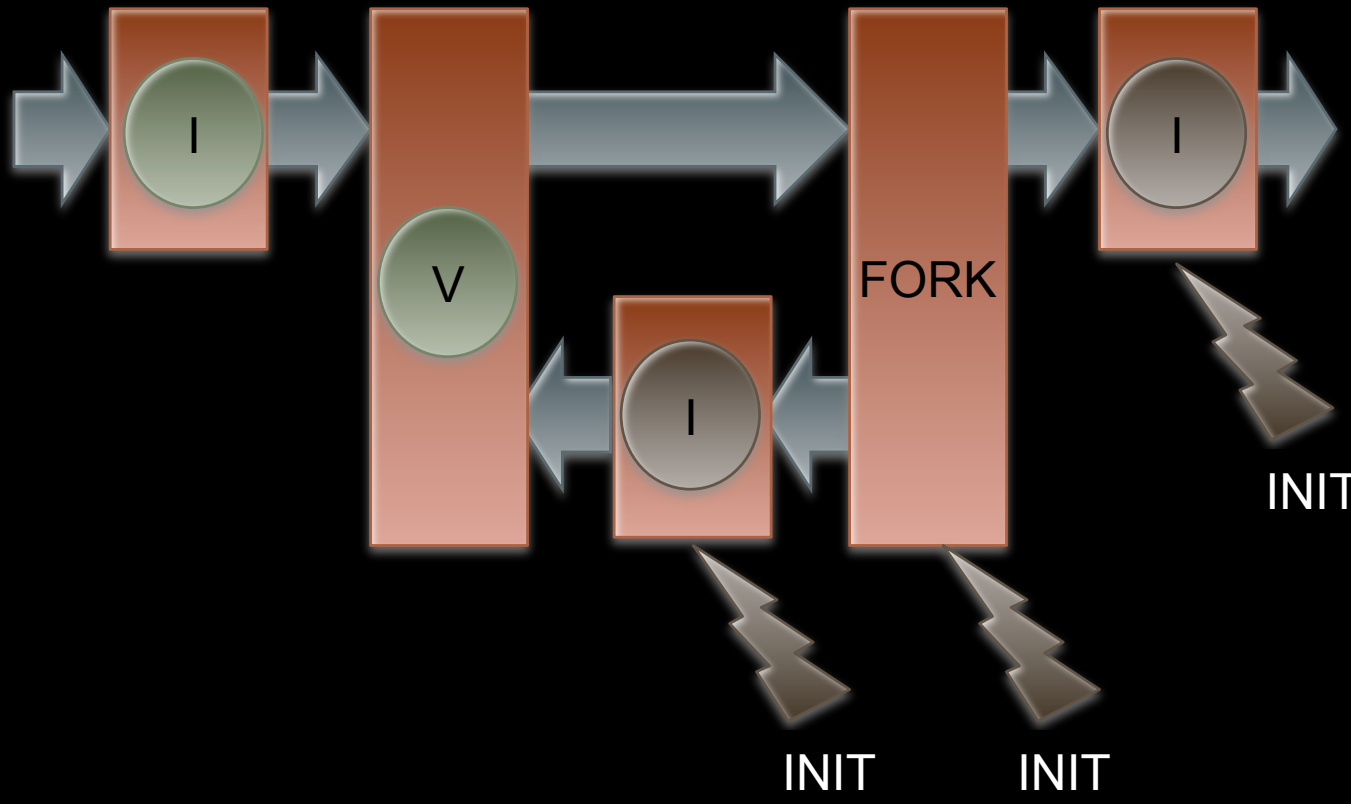
Deadlock



jeremie.hamon@imag.fr

Architecture des circuits QDI

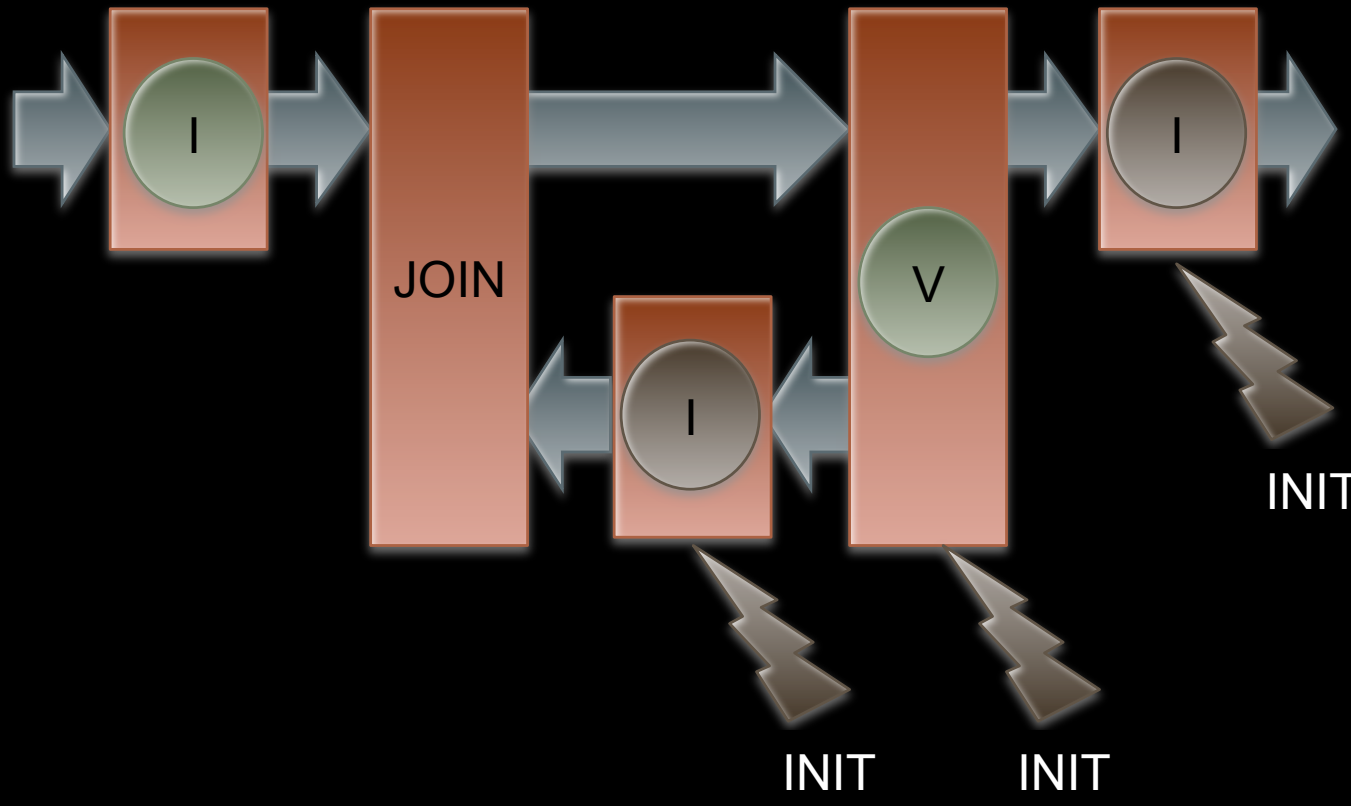
Deadlock



jeremie.hamon@imag.fr

Architecture des circuits QDI

Deadlock



jeremie.hamon@imag.fr

Architecture des circuits QDI

Prévention des deadlocks

- Il faut initialiser correctement les boucles pour
  - Prévenir les deadlocks
  - Respecter le protocole de communication
- Il faut au moins 3 HalfBuffers (ou équivalent) dans une boucle
  - 1 pour stocker le jeton valide
  - 1 pour stocker le jeton invalide
  - 1 pour une bulle

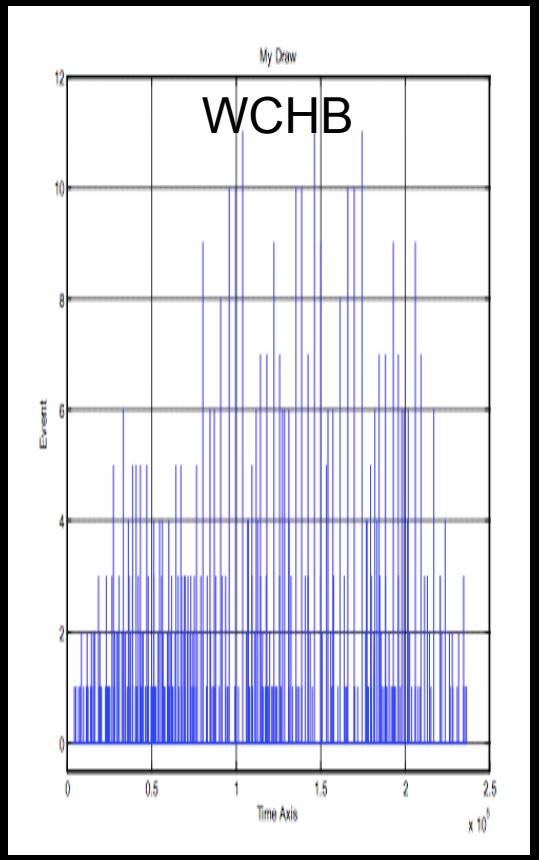
Conclusion

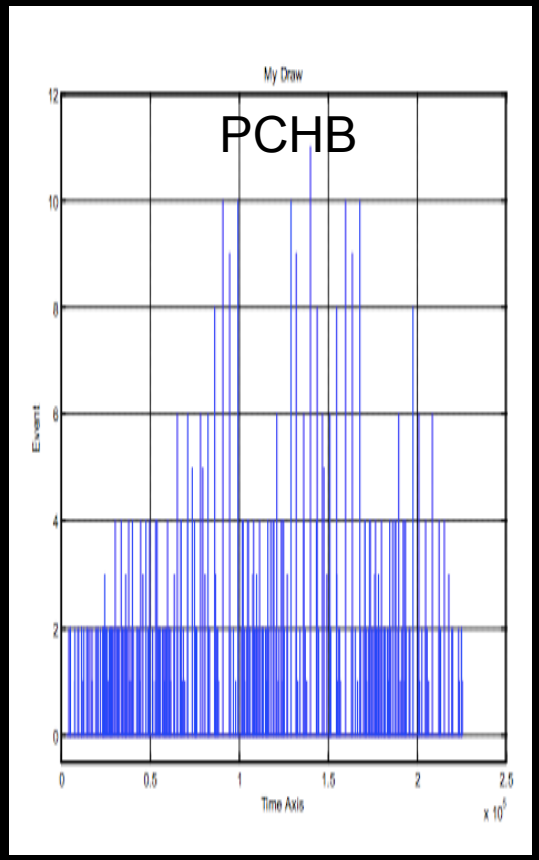
Conception en logique asynchrone

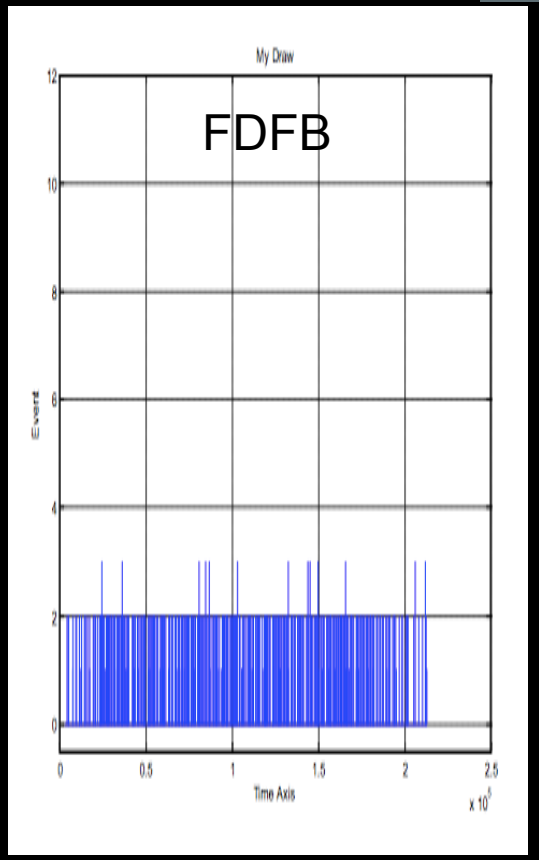
- Plusieurs styles d'implémentation
  - Portes de Muller : Dynamic, Weak Feedback, Conventional, Symetric
  - Buffers : Half Buffer, Full Buffer
  - Protocoles : WCHB, PCHB, FDFB
  - Acquittements : groupés, séparés, retardés
- Pour plusieurs caractéristiques différentes :
  - Consommation, Surface, Performance, EMI, Robustesse, Securité
- Grande finesse d'exécution!

Conclusion

Exemple : effet du protocole sur l'activité / EMI



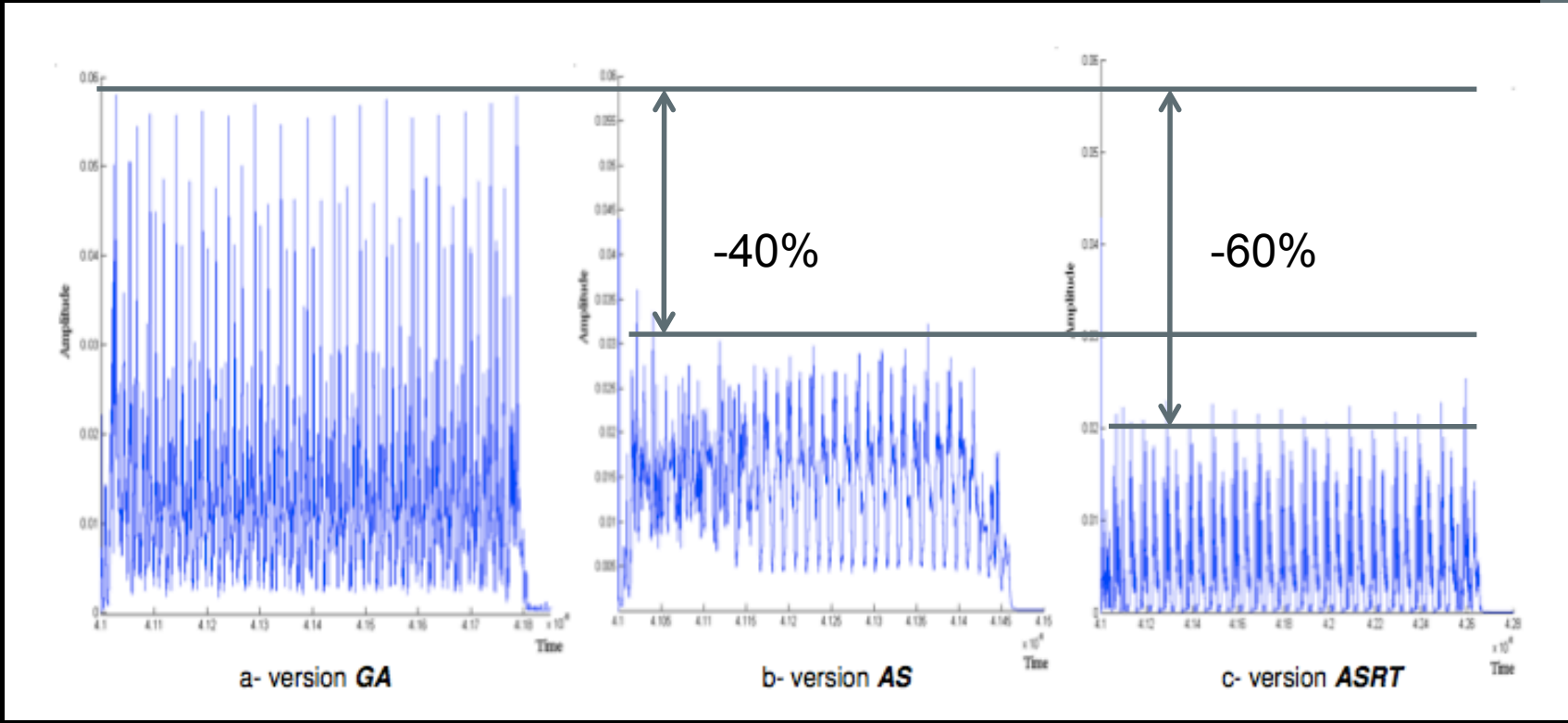




[YAHYA 2009] : Eslam Yahya, « Performance Modeling, Analysis and Optimization of Multi-Protocol Asynchronous Circuits », Thèse, Grenoble-INP

Conclusion

Exemple : effet de l'acquittement sur l'activité / EMI



[YAHYA 2009] : Eslam Yahya, « Performance Modeling, Analysis and Optimization of Multi-Protocol Asynchronous Circuits », Thèse, Grenoble-INP

Conclusion

Outils de conception

- Outils universitaires :
  - Petrify
  - Minimalist
  - Balsa
  - CAST
  - TAST
- Outils commerciaux :
  - Handskake Solutions
  - Achronix (FPGA asynchrones)
  - Tiempo (Startup de TIMA)

Merci de votre attention.

jeremie.hamon@imag.fr