TELECOM Paris

IP PARIS

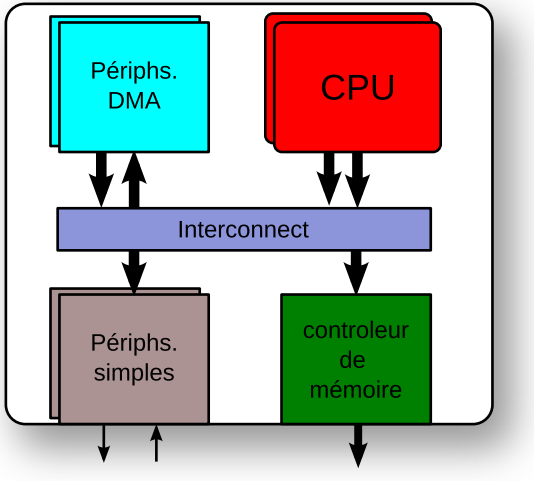
Introduction à la conception de systèmes sur puce

Interconnexion dans un système sur puce

Tarik Graba
tarik.graba@telecom-paris.fr
2024-2025

Interconnexion dans un SoC

Communication dans une puce



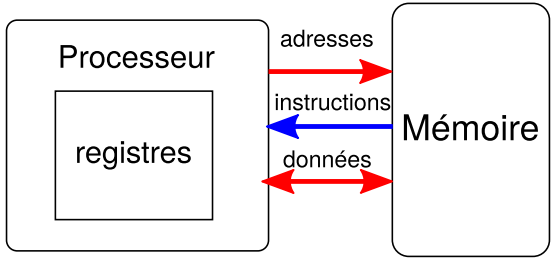
SoC

Dans un SoC, nous avons :

- Un ou plusieurs processeurs
- Des mémoires
  - internes
  - externes à travers un contrôleur
- Des périphériques
  - simples
  - qui peuvent accéder à la mémoire (DMA)

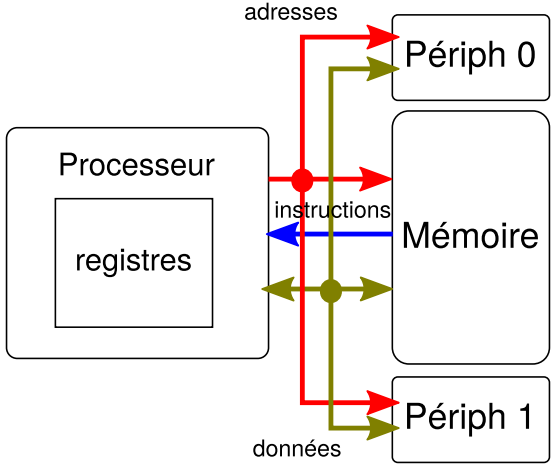
Périphériques mappés

Pour un processeur les données et instructions ont une adresse en mémoire.



Processeur/mémoire

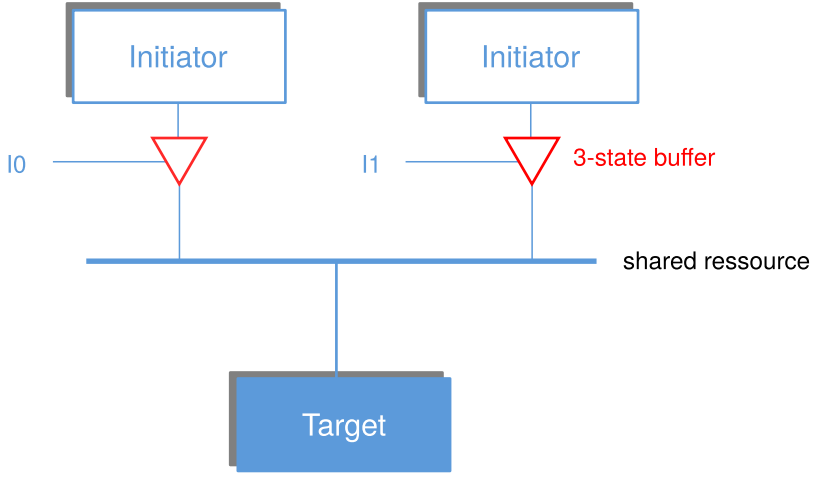
Dans un SoC moderne, les périphériques sont aussi mappés en mémoire.



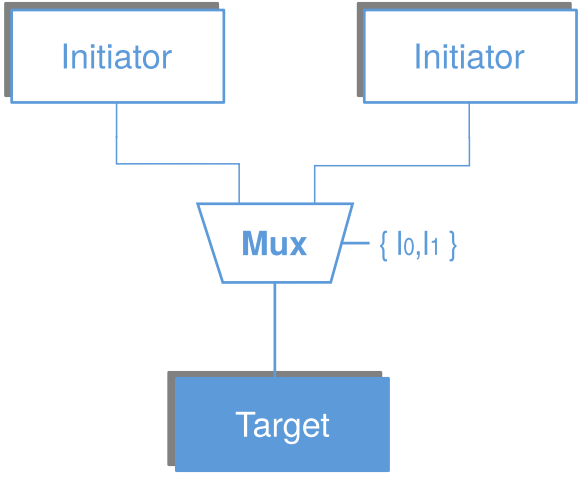
Processeur/mémoire/périphériques

Bus partagé

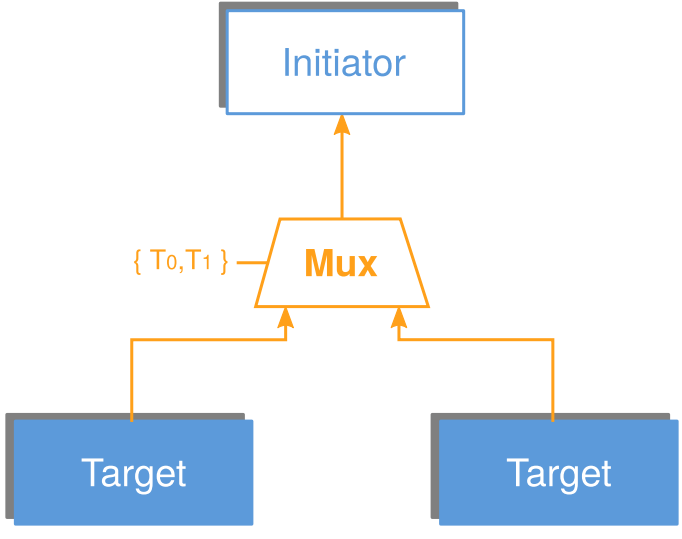
En utilisant des buffers 3-états



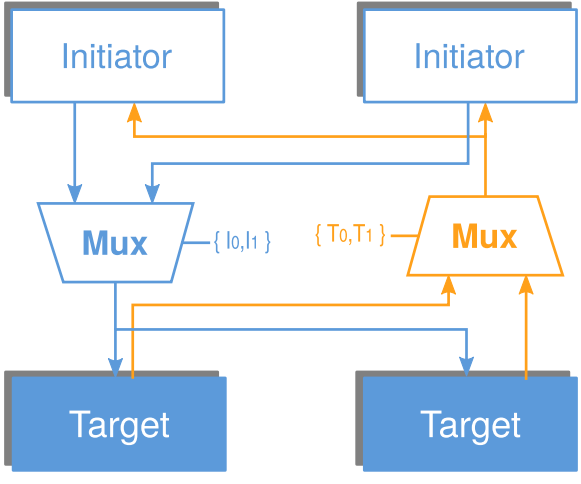
Bus multiplexé



multiplexed bus

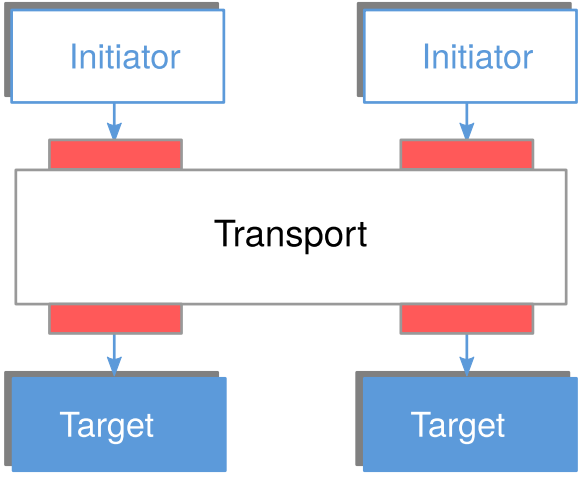


multiplexed bus



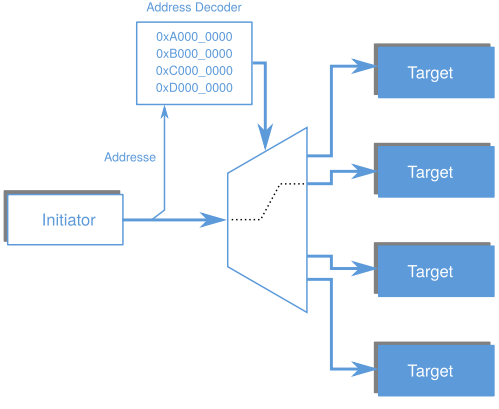
bidir multiplexed bus

Bus fabric

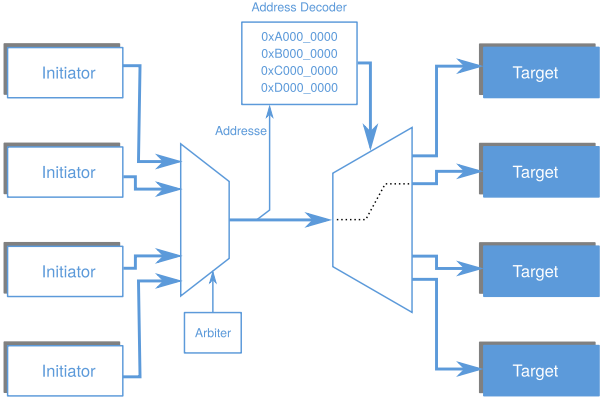


bus fabric

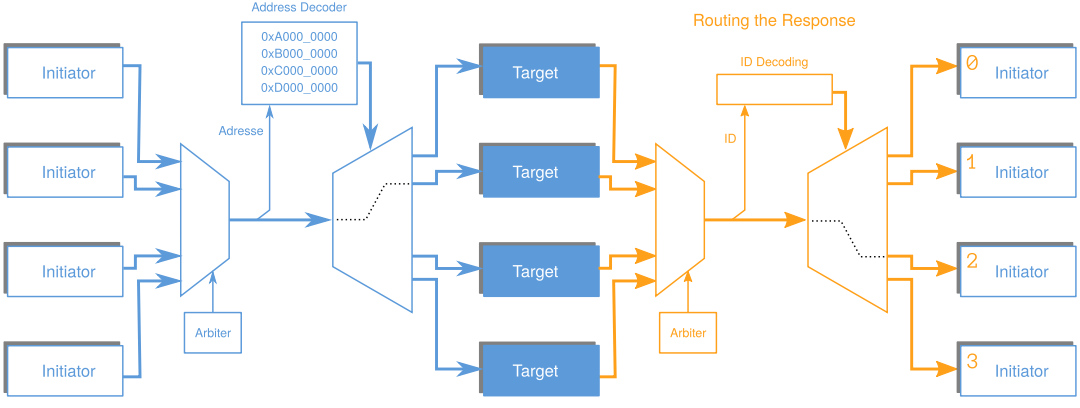
Fonctionnalités de base d'un interconnect



Interconnect

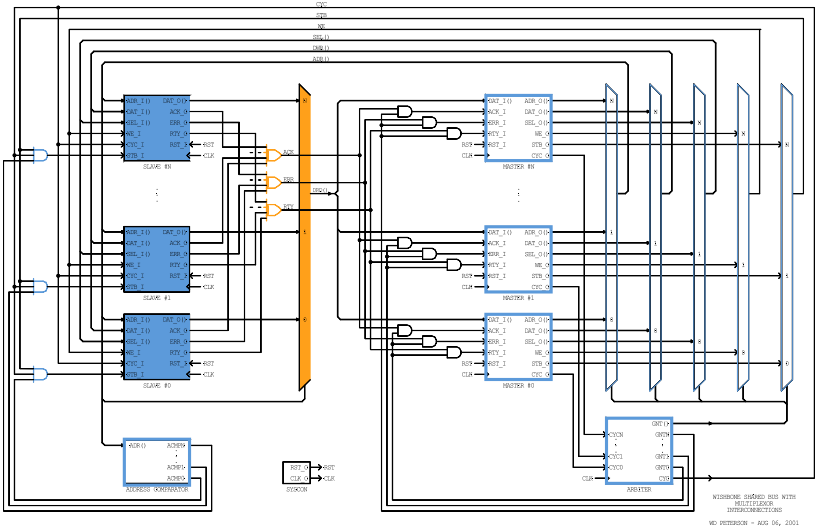


Interconnect



Interconnect

Exemple d'interconnect



Wishbone Interconnect

Protocoles de communication sur puce

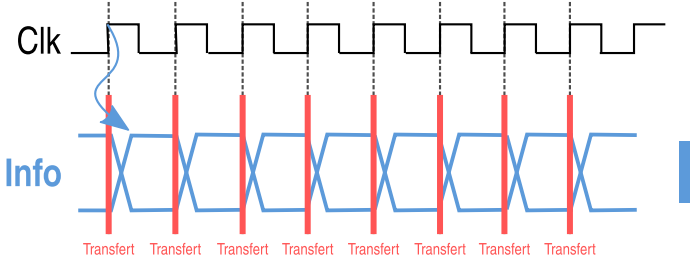
Standardisation des protocoles

- Protocoles standards pour garantir l'interopérabilité
  - Ensemble de fonctionnalités pour assurer la communication entre initiateurs et cibles
  - Utilisable pour différents niveaux de performances
  - Permettre des communications à faibles latences et/ou haut débits
- Protocoles point à point
  - Définir les transactions entre un initiateur et une cible
  - Ne pas dépendre de la topologie de l'interconnect
  - Comment s'interfacer à l'interconnect
- Utilisable dans un contexte de périphériques mappés en mémoire

Protocoles synchrones

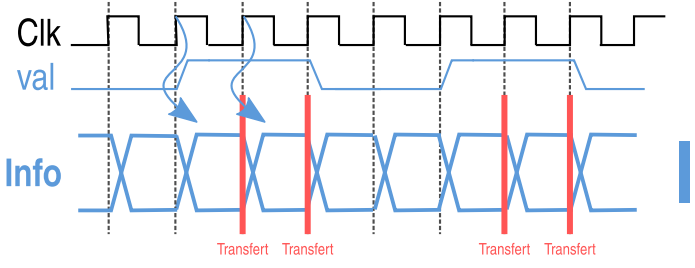
Pour avoir des garanties de performances, il faut un protocole synchrone.

- les sorties des initiateurs changent au front d'horloge
- les cibles capturent au front d'horloge



Protocole synchrone

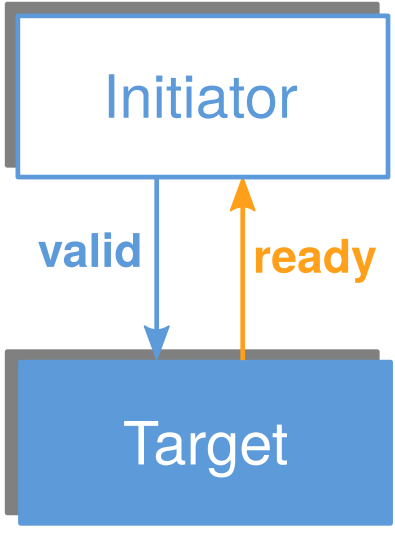
Signalisation



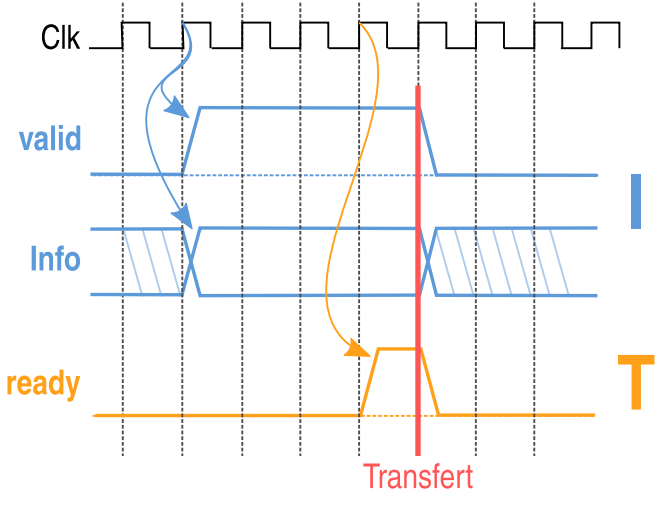
Protocole synchrone-signalisation

Rendez-vous

La cible peut mettre en attente l'initiateur (*back-pressure*).
Le transfert a lieu quand il est validé par l'initiateur et que la cible est prête à l'accepter.
On parle de rendez-vous (*handshake*).

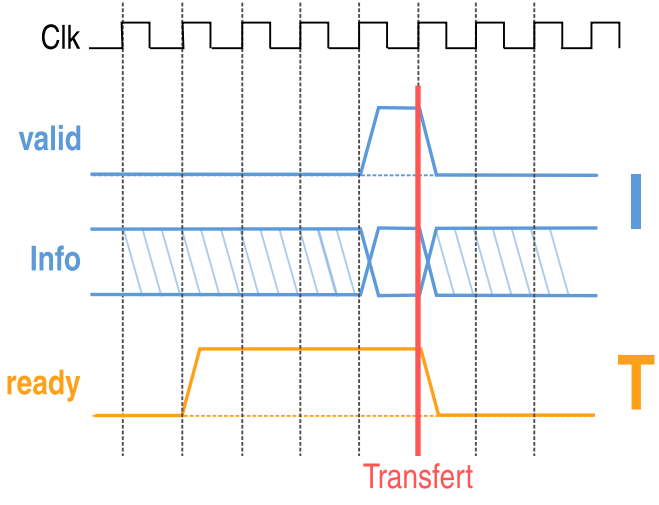


Rendez-vous



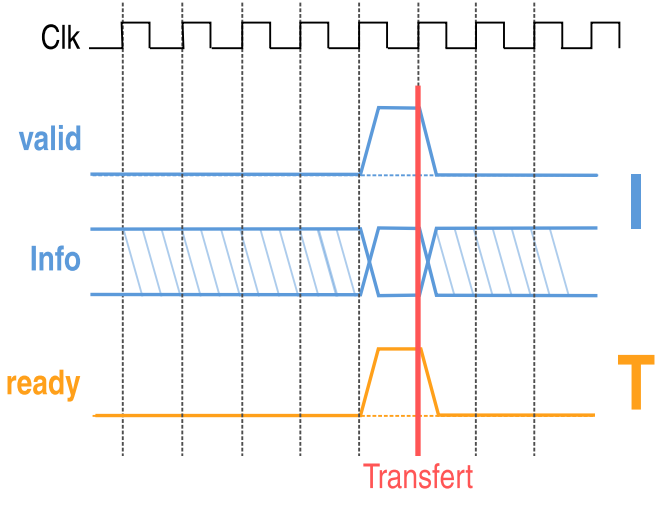
Rendez-vous

La cible peut faire attendre l'initiateur.



Rendez-vous

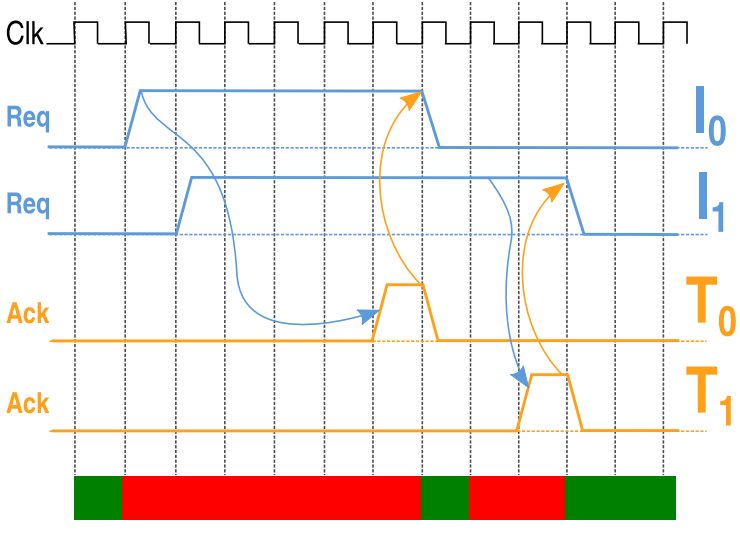
La cible peut être prête à l'avance.



Rendez-vous

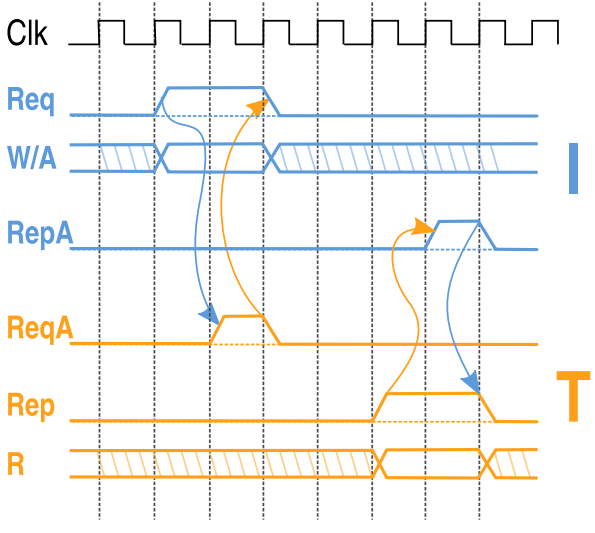
La cible et l'iniiateur peuvent être prêts dans le même cycle.

Transactions bloquantes

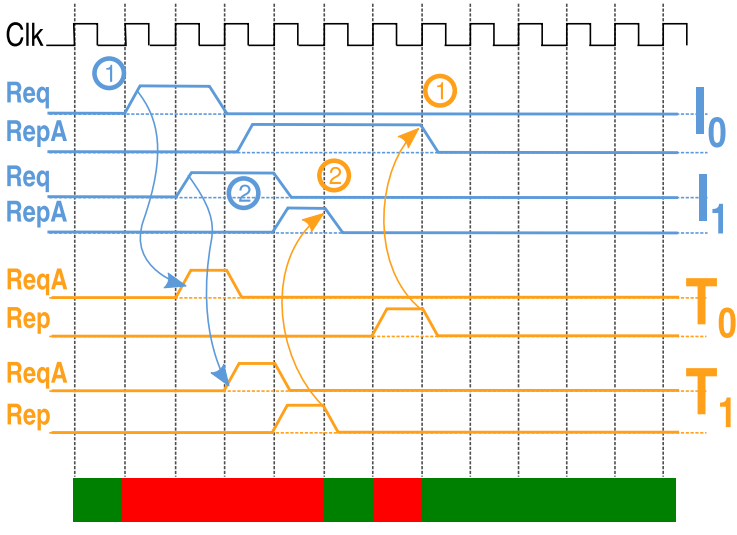


Transaction bloquante

Séparation des requêtes/réponses

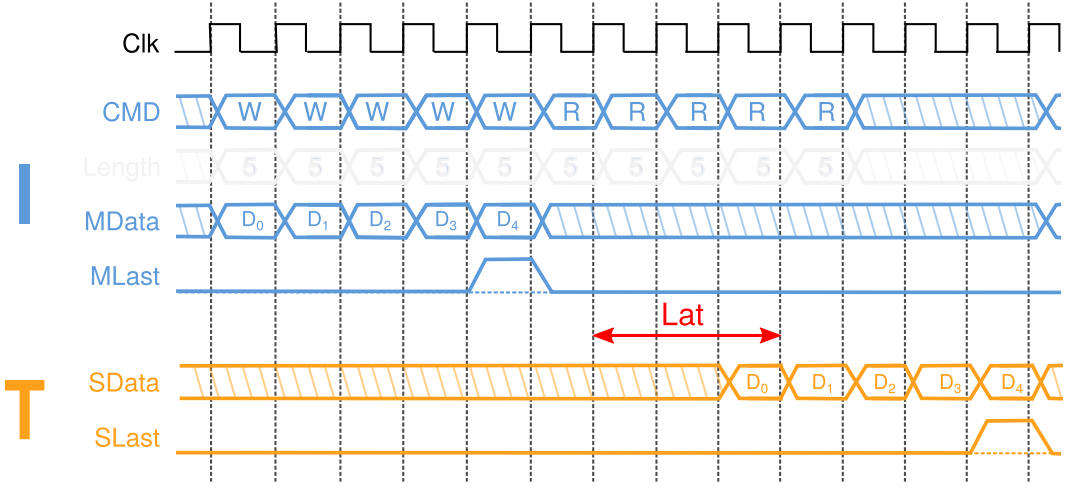


Transaction décomposée

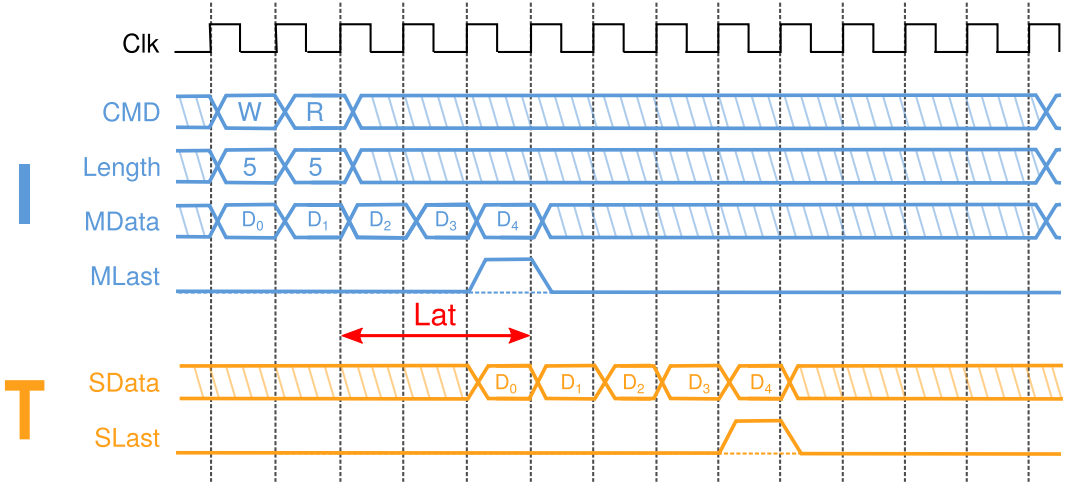


Transaction décomposée

Accès multiples (bursts)



Transactions multiples



Transactions uniques

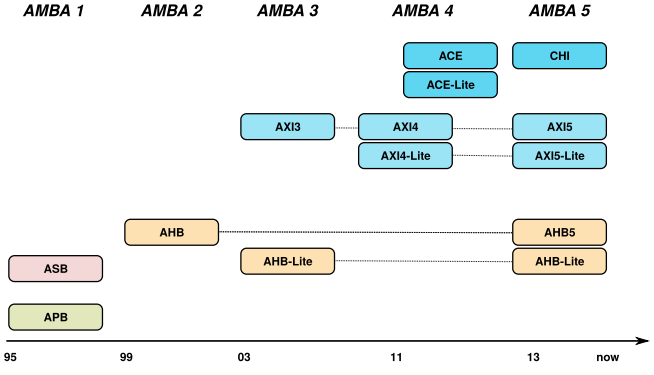
Comparatifs de différents protocoles

Quelques protocoles communs :

	rdv. synch	Req/Resp	SRMD
ARM AXI	+	+	+
ARM AXI-Lite	+	+	-
OpenCore OCP	+	+	+
ARM AHB/APB	+	-	+
Wishbone	+	-	-
Altera Avalon	+	pipeline	burst ext.

Le protocole AXI

AXI : Advanced eXtensible Interface



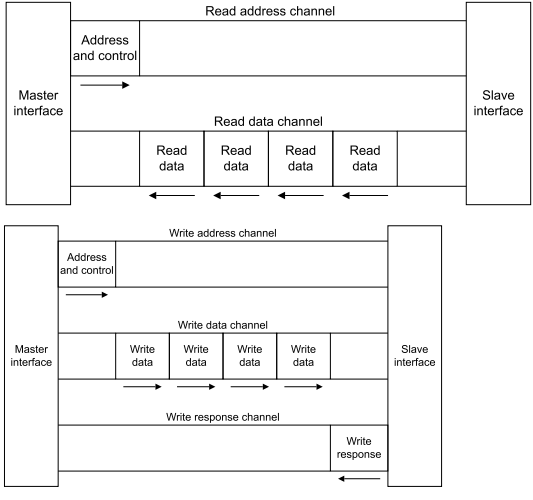
Historique AMBA

- AMBA : Advanced Microcontroller Bus Architecture
- Initialement propriétaire ARM, devenu un standard de fait
- Plusieurs évolutions pour s'adapter aux besoins des SoC modernes

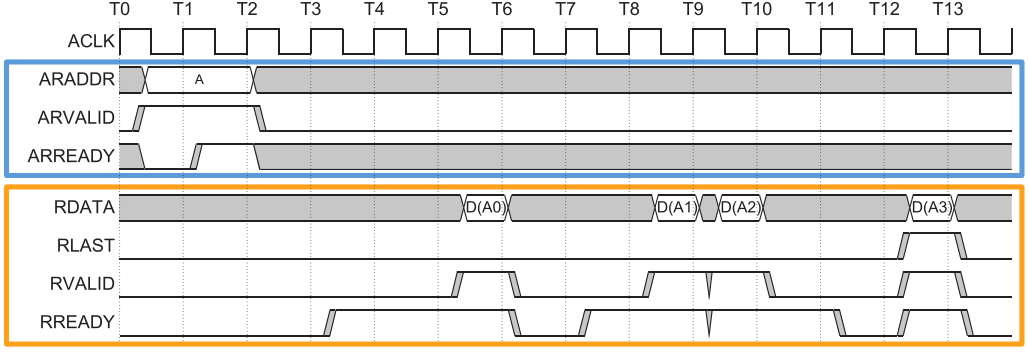
AXI : Caractéristiques

- Lecture et écriture séparés
- Requêtes et réponses séparées
- Conçu pour des SoC multiprocesseur avec de forts besoins en débit
  - Accès multiples (bursts)
  - Plusieurs maîtres/threads
  - Mode spéciaux (permission, caches…)

La spécification est publiquement disponible [lien].

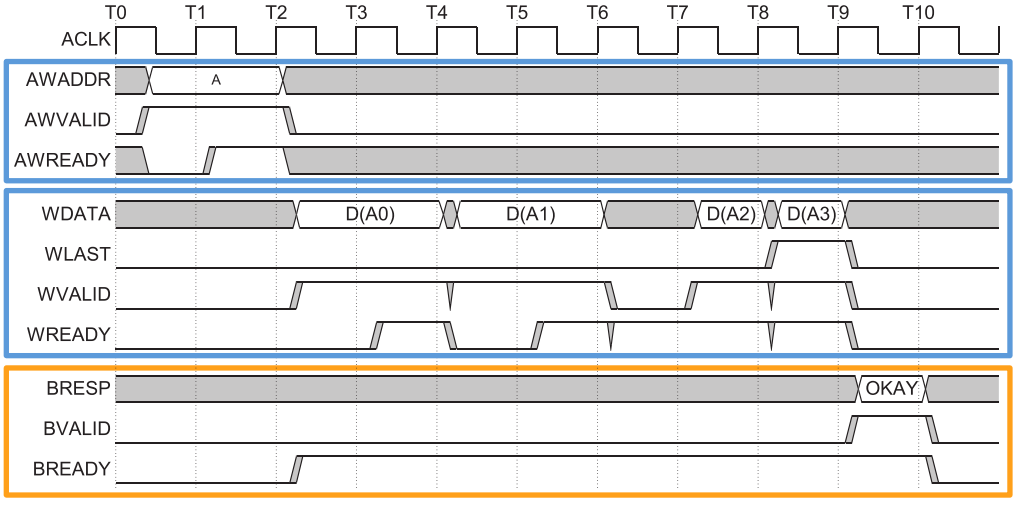


AXI : Exemple de lecture



Extrait de *AMBA AXI Protocol v2.0 (2010)*

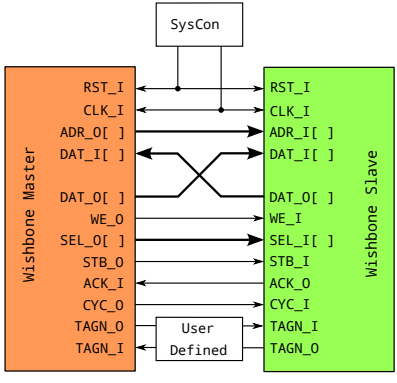
AXI : Exemple d'écriture



Extrait de *AMBA AXI Protocol v2.0 (2010)*

Le protocole Wishbone

Le protocole Wishbone



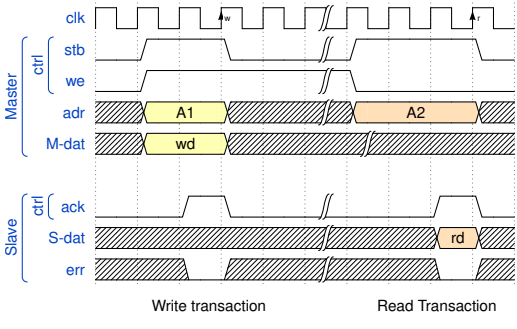
Interface Wishbone

Protocole de bus libre (Open source)
- La spécification officielle

À l'origine, liée aux projets OpenCores.org
- https://opencores.org/
  - plus très actif

Communication point à point entre *Master* et *Slave*

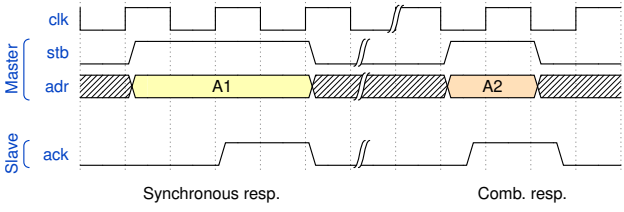
Transactions Wishbone

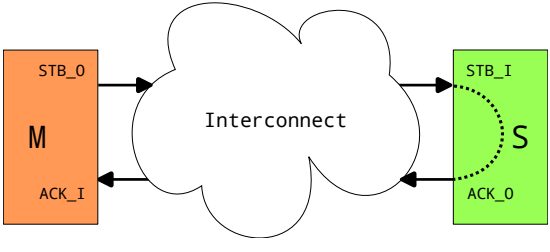


Rendez-vous synchrone entre `stb` et `ack`

- une transaction commence quand `stb` passe à 1
  - l'adresse doit être valide en même temps
  - we indique que c'est une requête d'écriture
- maintenu tant qu' `ack` ne passe pas à 1
- Des signaux en plus pour gérer les octets dans un mot, d'éventuelles erreurs, et autres signaux de contrôle.

Défaut de Wishbone





Le ack est vu comme une réponse à `stb` et le protocole autorise une réponse combinatoire.

- On peut se retrouver avec des boucles combinatoires sur tout l'interconect
  - Ce qui rend complexe l'estimation de timing

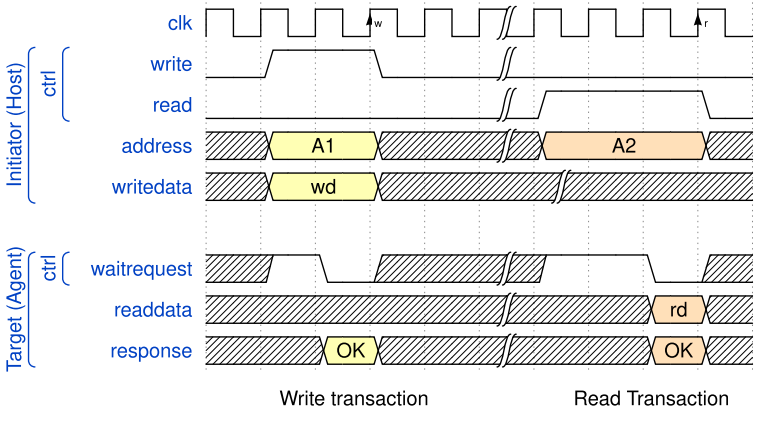
Le protocole Avalon

Avalon : Caractéristiques

- Développé par Altera (Intel) pour des SoC sur FPGA
- Pour des systèmes Memory Mapped
  - Existe des versions pour les flux (stream)
- Des extensions pour le Pipeline et les Burst
- Utilisé principalement pour les IPs Altera et les cœurs de processeur Nios

La spécification est publiquement disponible [lien].

Avalon : les transactions simples



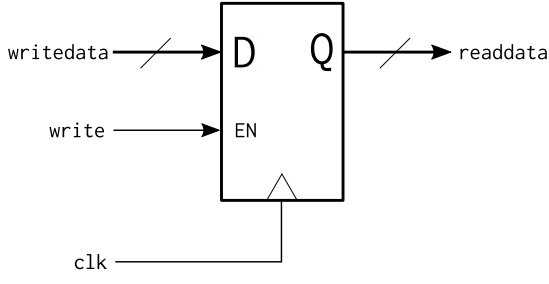
Simple transactions

Avalon, Exemple

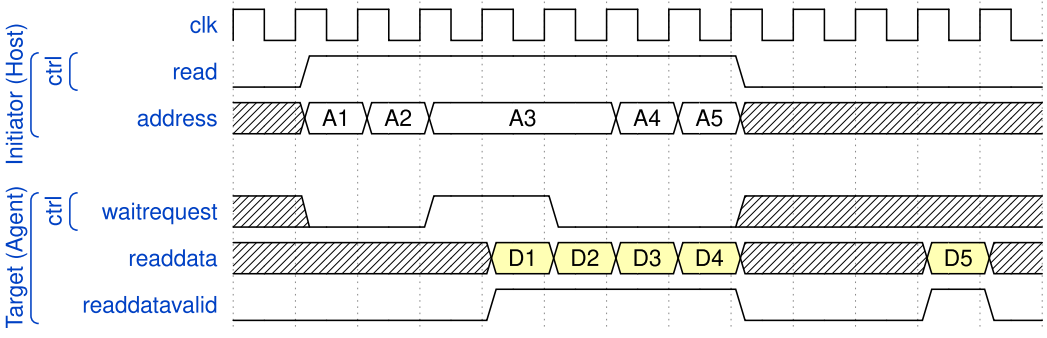
Les signaux de contrôle ne sont pas tous obligatoires.
Par exemple, pour un simple registre :

- `waitrequest` n'est pas utile, les données en sortie d'un registre sont toujours prêtes.
- `read` n'est pas utilie non plus.
- …

Ceci est un Agent Avalon (Cible) valide

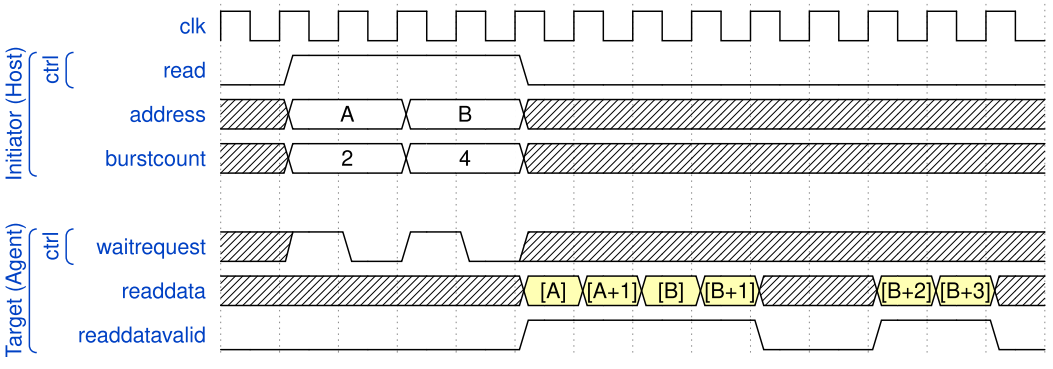


Avalon, lecture piplinée (extension)



Pipelined Read transactions

Avalon, bursts (extension)



Burst Read transactions