Bus Avalon

1. Définition

Avalon est un bus informatique destiné à être implémenté dans du matériel programmable (FPGA) développé par Altera. Ce bus est une architecture de bus simple conçue pour connecter des processeurs et des périphériques sur puce dans un système programmable sur puce (SOPC).

2. Objectifs de conception du bus Avalon

Les principaux objectifs de conception du bus Avalon étaient :

■ Simplicité:

Fournir un protocole facile à comprendre avec une courte courbe d'apprentissage.

■ Utilisation optimisée des ressources pour la logique du bus:

Conserver les éléments logiques (LEs) à l'intérieur du dispositif logique programmable (PLD).

■ Fonctionnement synchrone :

S'intègre bien avec d'autres logiques d'utilisateur qui coexistent sur le même PLD. qui coexiste sur le même PLD, tout en évitant les problèmes complexes d'analyse de synchronisation.

3. Principes de l'architecture Avalon

1. Les interfaces périphériques sont synchrones avec l'horloge Avalon. Par conséquent, il n'est pas nécessaire d'utiliser des schémas d'échange et de confirmation complexes et asynchrones. Complexe et asynchrone uniquement requis.

2. Tous les signaux sont actifs BAS ou HAUT et peuvent inverser le bus à la volée. Renversement immédiat du bus. Un multiplexeur (pas un tampon à trois états) sur le bus Avalon détermine quels signaux pilotent quels périphériques. Avalon détermine quels signaux pilotent quels appareils. L'appareil ne force pas les sorties à trois états, même si l'appareil n'est pas sélectionné. L'appareil est désélectionné.

3. Les signaux d'adresse, de données et de contrôle utilisent des ports dédiés séparés, ce qui simplifie la conception de l'appareil. L'appareil n'a pas besoin de décoder les cycles d'adresse

et de bus de données ou de désactiver les sorties lorsqu'elles ne sont pas sélectionnées. Si elle n'est pas sélectionnée, désactivez cette sortie.

4. Communication maitre-esclave

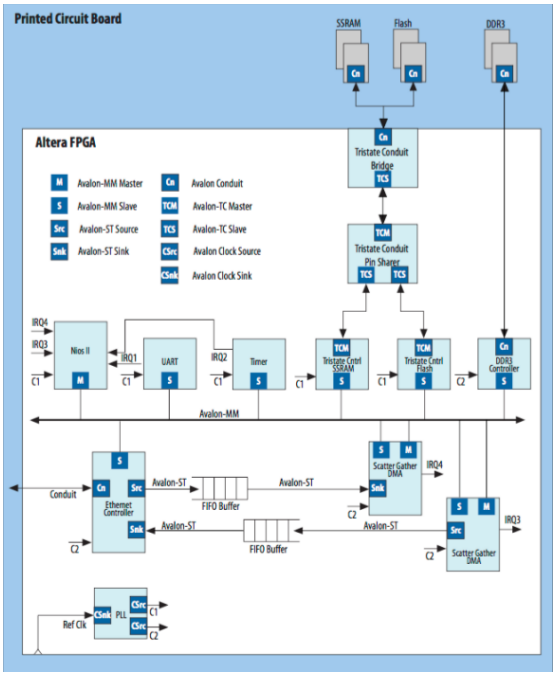
Avalon est un système de communication maître-esclave :

- Un composant Maitre sur le bus initie les transactions soit en envoyant directement des données, soit en émettant des requêtes aux composants esclaves.
- Un composant Esclave ne prend jamais l'initiative d'utiliser le bus (en lecture ou en écriture). Il ne fait que répondre aux requêtes des maitres.

5. Interfaces

On trouve comme interfaces :

- Avalon ST (Streaming Interface) → Communication unidirectionnelle de données.
- Avalon MM (Memory Map) → interface bidirectionnelle typique de connexion Maître/ Esclave. composants mappés dans l'espace mémoire (UART, microprocesseur, mémoires, DMA…).
- Avalon Conduit → permet l'ajout de signaux qui ne sont pas compris dans le standard (utile lors de la connection vers d'autres modules).
- Avalon TC (Tri-State Conduit) → permet la connexion à des périphériques off-chip.
- Avalon Interrupt / Clock / Reset → permet le partage de signaux d'interruption / horloge /reset.



6. Adressage

Pour l'esclave :

- L'adresse en entrée des ports esclaves est une adresse de mots désignant donc un offset dans l'espace d'adressage du port esclave.

-Chaque adresse accède donc à un mot complet par rapport à la largeur des signaux readdata ou writedata.

Pour le maître :

- Les adresses envoyées par le maître sont des adresses d'octets, sans prise en compte de la largeur des bus de données.

-Par exemple, un port maître de données de 32 bits devra aligner ses adresses sur des frontières de 4 octets : 0x00, 0x04, 0x08, 0x0C…

-Pour accéder à un octet spécifique dans un mot, le maître doit utiliser le signal byteenable.

7. Registres

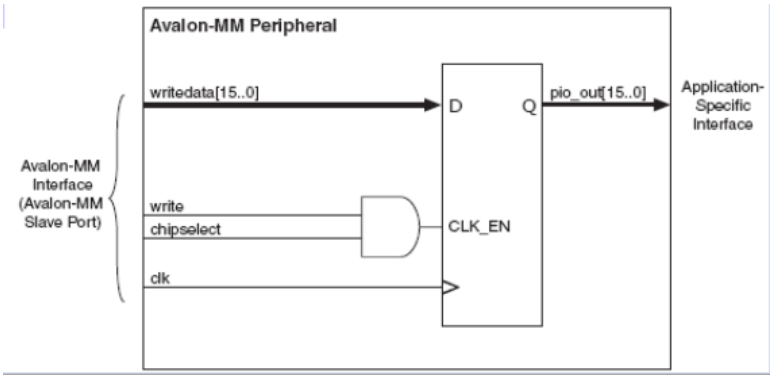
Un registre :

-writedata,

-write,

-chipselect,

-clk



a) Transfert en mode esclave:

- Comme l'adresse d'octet est comprise comme une adresse de mot par l'esclave, le maître envoie le signal Byteenable.
- C'est un vecteur dont la taille est égale au nombre d'octets dans la largeur du bus de données.
- Sans sa présence, l'esclave renverrait tout le mot.
- Les combinaisons valides

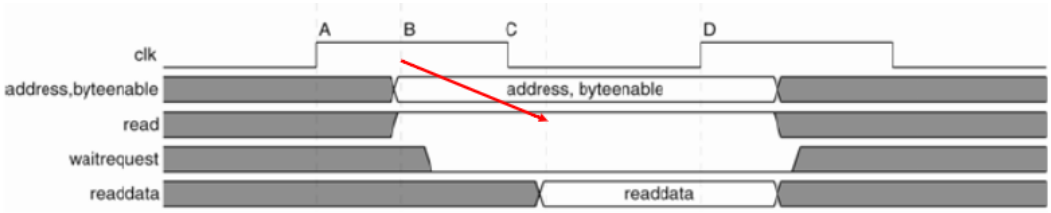
  En 32 bits sont :

  0001, 0010, 0100, 1000, 0011, 1100, 1111

Table 4: Byte-Enable Example for a 32-Bit Slave Port

Byteenable [3..0]	Write Action
1111	Write full 32-bits
0011	Writes lower 2 bytes
1100	Writes upper 2 bytes
0001	Write byte 0 only
0100	Write byte 2 only

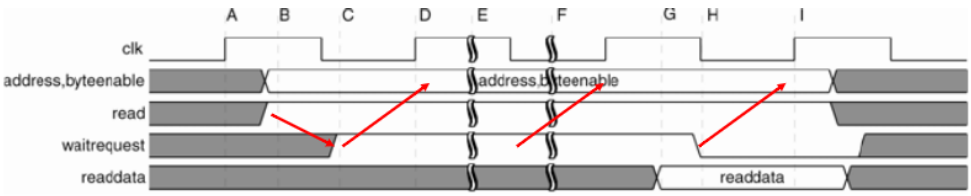
b) Lecture en mode Maitre en 1 cycle



Notes to Figure 14:

(A) First cycle starts on the rising edge of `clk`.
(B) Master port asserts valid `address`, `byteenable` and `read`.
(C) Valid `readdata` returns from the system interconnect fabric during first cycle.
(D) Master port captures `readdata` on the next rising edge of `clk` and deasserts all its outputs. The read transfer ends here and the next cycle could be the start of another transfer.

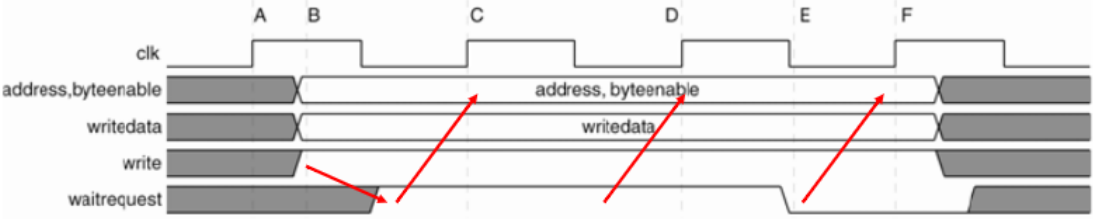
c) Lecture en mode Mitre avec cycles d'attente



Notes to Figure 15:

(A) First cycle starts on the rising edge of `clk`.
(B) Master asserts valid `address`, `byteenable` and `read`.
(C) System interconnect fabric asserts `waitrequest` before the next rising edge of `clk`.
(D) Master port accepts `waitrequest` at the rising edge of `clk`. This cycle becomes a wait-state.
(E-F) As long as `waitrequest` is asserted, master port holds all outputs constant.
(G) Valid `readdata` returns from system interconnect fabric.
(H) System interconnect fabric deasserts `waitrequest`.
(I) Master port captures `readdata` on the next rising edge of `clk` and deasserts all outputs. The read transfer ends here, and the next cycle could be the start of another transfer.

d) Principe en mode Maitre avec cycles d'attente



Notes to Figure 17:

(A) First cycle starts on the rising edge of `clk`.
(B) Master asserts valid `address`, `writedata` and `write`.
(C) `waitrequest` is asserted at the rising edge of `clk`, so this cycle becomes the first wait-state. Master holds all outputs constant.
(D) `waitrequest` is asserted at the rising edge of `clk` again, so this becomes the second wait-state. Master holds all outputs constant.
(E) System interconnect fabric deasserts `waitrequest`.
(F) `waitrequest` is not asserted at the rising edge of `clk`, so master deasserts all outputs, and the write transfer terminates. Another read or write transfer may follow on the next cycle.