ELE8307 - Laboratoire 4
Conception de périphériques dédiés pour le système d'interconnexions Avalon

Introduction

Lors de la conception d'un système embarqué, on cherche généralement à utiliser le matériel à disposition pour accélérer les traitements. Il est fréquent d'avoir à adapter un périphérique à l'interface d'un système déjà en place. De plus, lorsque le processeur utilisé n'est pas extensible (par instructions spécialisées par exemple), il faut concevoir un coprocesseur pour accélérer les calculs avec du matériel spécialisé. Il s'agit en général d'un périphérique fonctionnant en « maître/esclave », que l'on relie au système. Il faut donc connaître l'interface utilisée : dans notre cas, le système d'interconnexion Avalon.

Dans ce laboratoire, nous produirons deux périphériques spécialisés qui pourront ensuite être intégrés facilement au système avec Qsys. Dans un premier temps, afin de permettre à l'usager de se déplacer dans le système solaire, nous utiliserons le contrôleur clavier réalisé au laboratoire 1. Dans un deuxième temps, nous produirons un contrôleur VGA avec double tampon mémoire afin d'offrir un affichage de meilleure qualité.

Objectifs

À la fin de ce laboratoire, l'étudiant aura développé différentes compétences liées à la conception de périphériques spécialisés ciblant un système d'interconnexion particulier:

- ✓ Compréhension des transferts de base du système d'interconnexion Avalon.
- ✓ Conception de périphériques spécialisés pour le système d'interconnexion Avalon.

Documentation

La documentation de référence pour le système d'interconnexion Avalon est disponible sur Moodle.

Introduction

L'utilisation du système d'interconnexion Avalon simplifie grandement le travail de connexion des composants d'un système complexe. La spécification Avalon définit 6 types d'interface :

- Avalon Memory-Mapped Interface (Avalon MM): Une interface basée sur l'écriture/lecture à différentes adresses du système.
- Avalon Streaming Interface (Avalon ST): Une interface supportant un débit unidirectionnel de données pour les applications requérants une bande-passante importante.

- Avalon Memory Mapped Tristate Interface: Une interface basée sur l'écriture/lecture à différentes adresses pour les périphériques à l'extérieur de la puce. En partageant les bus d'adresses et de données, il est possible de réduire le nombre de broches utilisées sur le FPGA.
- Avalon Clock Interface: Une interface de distribution d'horloge et du signal de réinitialisation.
- Avalon Interrupt Interface: Une interface qui permet de signaler des événements d'un composant à l'autre.
- Avalon Conduit Interface: Une interface qui permet de faire sortir/entrer des signaux de l'entité de haut niveau du système Qsys.

Avalon Memory-Mapped Interface

Dans un système produit avec Qsys, tous les composants tels que les processeurs, mémoires, timers et autres périphériques se voient assigner un espace d'adresses. La communication entre ces composants se fait alors via des lectures et écritures à ces adresses. Dans un tel système d'interconnexions, on distingue le maître, qui peut initier un transfert, de l'esclave qui ne peut que répondre à une requête. Des caractéristiques intéressantes de l'interface Avalon MM sont :

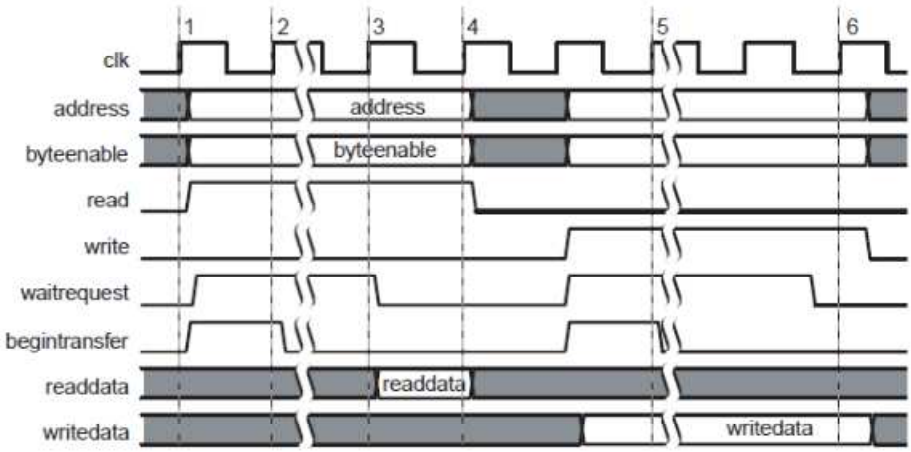
- Définition d'une connexion point-à-point entre un composant et un système d'interconnexion.
- Liberté d'implémenter seulement un sous-ensemble requis de signaux.
- Supporte des données de taille variables (8, 16, 32, 64, …, 1024 bits) pour différents composants.
- Génération automatique du tissu d'interconnexion.

Les esclaves

Pour les besoins de ce laboratoire, nous nous intéresserons plus spécifiquement aux transferts du point de vue de l'esclave, puisque les contrôleurs clavier PS/2 et VGA sont des esclaves. Néanmoins, les transferts du point de vue maître ne diffèrent que par peu de choses. La table suivante présente les principaux signaux de l'interface esclave :

Signal	Largeur	Direction	Description
read	1	Entrée	Indique le début d'un transfert de lecture. Si présent, on doit utiliser le signal « readdata » aussi.
write	1	Entrée	Indique le début d'un transfert d'écriture. Si présent, on doit utiliser le signal « writedata » aussi.
address	1-32	Entrée	Spécifie le décalage (d'adresse) dans l'espace adressable de l'esclave.
readdata	8, 16, …, 1024	Sortie	La donnée requise par le transfert de lecture.
writedata	8, 16, …, 1024	Entrée	La donnée à écrire dans un transfert d'écriture.
begintransfer	1	Entrée	Indique le début d'un transfert, sans égard au signal « waitrequest ». Mis à '1' pendant un cycle.
waitrequest	1	Sortie	Indique que l'esclave n'est pas en mesure de répondre à la requête pour le moment (demande d'attendre). Les signaux de l'interface demeurent constants, à l'exception de « begintransfer » et « beginbursttransfer ».

Voici un exemple typique de transferts d'écriture et de lecture pour un esclave, dans lequel une requête d'attente est mise en évidence (voir section 3.2 de la documentation):



Plutôt que d'utiliser le signal « waitrequest », il est également possible lorsque des temps d'attentes constants existent de les fixer lors de la génération du composant dans Qsys avec les propriétés « writeWaitTime » et « readWaitTime ».

Transferts pipelinés

Il convient de mentionner qu'il est possible de configurer un composant afin qu'il supporte des lectures pipelinées, ce qui permet d'augmenter le débit des transferts avec ce composant. Les transferts d'écriture ne peuvent pas être pipelinés. La latence du pipeline est contrôlée par le paramètre « readLatency ». Cette caractéristique ne sera pas exploitée dans ce laboratoire (sauf si vous insistez!).

Transferts en rafale

Il convient également de mentionner la possibilité de supporter les transferts en rafale (Burst). L'idée des transferts en rafale est de verrouiller l'arbitration du composant pendant un nombre de transferts déterminé. Dans certains cas, les transferts en rafales peuvent être utiles et augmentent le débit de transferts.

Alignement d'adresse

Le service d'alignement d'adresse dynamique (Dynamic Bus Sizing) est offert par le système d'interconnexion Avalon lorsque la taille des données de l'esclave et du maître diffère, de sorte que les données de l'esclave soient alignées par octet dans l'espace adressable du maître. Par exemple, lorsqu'un maître de 32bits effectue un transfert de lecture avec un esclave de 16bits, deux lectures à des adresses consécutives de l'esclave sont réalisées. La table suivante illustre ces cas par quelques exemples :

Master Byte Address *(1)*	32-Bit Master Data	
	When Accessing a 16-Bit Slave Port	When Accessing a 64-Bit Slave Port
0x00	OFFSET[1]$_{15..0}$:OFFSET[0]$_{15..0}$ *(2)*	OFFSET[0]$_{31..0}$
0x04	OFFSET[3]$_{15..0}$:OFFSET[2]$_{15..0}$	OFFSET[0]$_{63..32}$
0x08	OFFSET[5]$_{15..0}$:OFFSET[4]$_{15..0}$	OFFSET[1]$_{31..0}$
0x0C	OFFSET[7]$_{15..0}$:OFFSET[6]$_{15..0}$	OFFSET[1]$_{63..32}$
. . .	. . .	. . .

Générer un composant Avalon MM

La génération d'un composant Avalon MM est similaire à la génération d'instructions spécialisées. Dans Qsys, il suffit d'aller dans File → New Component… et la même interface de spécification apparaît. Dans l'onglet HDL Files on spécifie le fichier VHDL source et l'entité de haut niveau. Dans l'onglet « Signals » on spécifie les interfaces utilisées, et la différence est qu'on utilise une interface de type « Avalon Memory-Mapped Slave » plutôt qu'une interface « Custom Instruction Slave », les signaux sont donc différents. Également, il faut associer les signaux clk et rst_n à l'interface Clock. L'onglet « Interfaces » vous permet de configurer certaines propriétés de chaque interface utilisée. Pour ce laboratoire, vous devez faire attention de bien contrôler les différents waveform proposés. Lorsque la génération est complétée, vous pouvez ajouter le composant Avalon au système en le sélectionnant dans la liste comme pour tout composant dans Qsys. Vous pouvez également faire un clic-droit puis « Edit Component » pour éditer un composant que vous avez créé.

Travail à réaliser

Ce laboratoire est divisé en deux parties. Dans la première partie, vous partirez du bloc VHDL du contrôleur clavier PS/2 du laboratoire 1 pour le transformer en composant Avalon utilisant l'interface Memory-Mapped. En deuxième partie, vous devrez produire un contrôleur VGA à double tampon mémoire dans le but de remplacer le contrôleur VGA à tampon unique. Nous devrons réintégrer le contrôleur VGA au sein du système Qsys. Il serait donc pertinent d'en profiter pour enlever les instructions spécialisées avec l'interface externe vers ce contrôleur VGA.

Partie 1 – Contrôleur clavier PS/2

En principe, la logique opérative du contrôleur clavier est déjà fonctionnelle, avec en sortie une FIFO qui accumule les octets émis par le clavier. Ainsi il devrait être relativement aisé de produire un composant clavier PS/2 pour le système d'interconnexions Avalon. Toutefois, il faut modifier le code VHDL pour que lorsque la FIFO soit vide, le code -1 soit émis en sortie. Il suffit de rajouter un petit process pour faire cela. Veuillez noter que vous pouvez aussi modifier les caractéristiques de votre « lpm_fifo» **si nécessaire** en réutilisant le wizard (par exemple, regardez ce que fait le show-ahead). Également, votre déclaration d'entité devra être la suivante :

```
entity ctrl_clavier_avalon is
	port (
		ps2_clk		:	in std_logic;
		ps2_dat		:	in std_logic;

		data		:	out std_logic_vector(31 downto 0);
		rd		:	in std_logic;

		clk		:	in std_logic;
		rst_n		:	in std_logic
	);
end ctrl_clavier_avalon;
```

Lorsque votre composant est généré et intégré au système, écrivez une petite routine qui boucle à l'infini pour afficher les touches (le code) entrées au clavier. Il faudrait filtrer les break codes des touches ainsi que le code de la touche qui y est associée. Pour ce faire, utilisez les Scancode fourni dans le dossier html du premier laboratoire. Vérifiez que tout fonctionne correctement.

Inspirez vous du fichier source ss_win32_main.c pour supporter les déplacements de l'observateur dans l'espace avec les touches W, A, S, D, Q, E, ←, ↑, →, ↓, PageUp, PageDown. Réutilisez les mêmes appels de fonction. Il n'est pas obligatoire d'utiliser les touches spéciales (←, ↑, →, ↓, PageUp, PageDown), vous pouvez les remplacer par des touches normales.

Touche	**Déplacement**
W	Avancer
S	Reculer
A	Déplacement latéral à gauche
D	Déplacement latéral à droite
Q	Rotation à gauche (selon l'axe top)
E	Rotation à droite (selon l'axe top)
←	Rotation à gauche (selon l'axe dir)
→	Rotation à droite (selon l'axe dir)
↑	Rotation vers le bas (selon l'axe side)
↓	Rotation vers le haut (selon l'axe side)
Page Up	Déplacement latéral vers le haut
Page Down	Déplacement latéral vers le bas

Dans votre main(), vous aurez maintenant un appel à la fonction readClavier() que vous devrez implémenter, avant l'appel habituel à display(). Dans cette fonction, assurez-vous de boucler sur un certain nombre d'itération pour être sûr de bien lire toutes les touches enfoncées entre 2 appels de la fonction. Utilisez la commande suivante pour aller lire les touches enfoncées :

```
int key = IORD( CTRL_CLAVIER_AVALON_INST_BASE , 0 );
```

Vous pourrez ensuite analyser la touche et produire le résultat escompté.

Partie 2 – Contrôleur VGA avec double tampon mémoire

Un problème majeur avec l'utilisation d'un seul tampon mémoire est que lorsque l'affichage n'est pas généré assez rapidement, on peut alors voir les images se dessiner, ou alors un clignotement peut exister.

La mémoire utilisée comme tampon mémoire pour l'écran VGA est une SRAM de 256k*16bits. Considérant des pixels de 8bits, un tampon mémoire d'un écran en résolution 640x480 représente 307.2k*8bits, il serait donc impossible de produire 2 tampon écrans avec la mémoire disponible. Il serait toutefois possible de réduire la taille de l'écran à 576x432pixels, soit 248k*8bits.

Puisque la mémoire est structuré en mots de 16bits, les adresses 0 à 128k seront réservées au premier tampon mémoire, et les adresses 128k à 256k seront réservées au deuxième. Il sera ainsi plus rapide de réaliser la lecture mémoire de pixels pour l'affichage d'un tampon. Comme dans le contrôleur VGA simple, vous utiliserez une FIFO d'entrée, pour recevoir les pixels à écrire en mémoire, et une FIFO de sortie pour les pixels lus en mémoire devant être affiché à l'écran.

Adresse mémoire SRAM	8bits hauts	8bits bas
0 (début tampon 1)	Pix_1	Pix_0
1	Pix_3	Pix_2
…	…	…
124415 (fin du tampon 1)	Pix_248831	Pix_248830
..	X	X
131071	X	X
131072 (début tampon 2)	Pix_1	Pix_0
131072+1	Pix_3	Pix_2
…	…	…
131072+124415 (fin du tampon 2)	Pix_248831	Pix_248830
…	X	X
262143	Réservé	Réservé

Fonctionnement

On désire que les deux tampons se relaient : lorsque l'un est en mode écriture, l'autre est en mode lecture pour affichage VGA. Ainsi on ne voit jamais l'image se dessiner à l'écran comme c'est le cas avec un seul tampon. De plus, lorsque l'usager tente d'écrire à l'adresse 262143, les tampons sont inter-changés (swap buffer), de sorte que si l'on lisait au départ sur le tampon 1 et écrivait sur le tampon 2, après le changement on écrit sur le

tampon 1 et on lit sur le tampon 2. Mais idéalement lorsqu'on commence à dessiner sur un tampon, il serait préférable que celui-ci ait déjà été effacé préalablement. Ainsi, lorsqu'il y a un changement de tampon, avant de réaliser une écriture supplémentaire à partir de la FIFO d'entrée, il faudrait automatiquement effacer le nouveau tampon d'écriture. Dans la description HDL, il faut spécifier une constante pour la couleur d'effacement (fixée à noir). Il faut prendre bien soin de synchroniser la lecture mémoire avec le début d'une nouvelle image à l'écran.

Il est également souhaité que pour le programmeur logiciel, l'écriture des pixels 0 à 248831 se fassent par des demandes d'écritures aux adresses 0 à 248831. La conversion des adresses reçues à la FIFO d'entrée en adresses de la mémoire en fonction du tampon d'écriture courant doit être gérée par le contrôleur.

Les ports de l'entité VGA2 seront les mêmes que ceux du contrôleur VGA à simple tampon mémoire. Afin de modifier la taille de l'écran, il faut modifier légèrement la génération des signaux video_on_h et video_on_v. On veut que l'écran soit centré et que le cadre soit affiché en noir.

La mémoire SRAM

La mémoire SRAM possède des ports de données bidirectionnels, il sera donc très important de faire attention lors de la conception et de la simulation du contrôleur afin d'utiliser des tristate buffers en mode haute impédance lors de la lecture mémoire.

Les signaux de contrôles de la mémoire sont CE, WE, OE, LB, et UB, et fonctionnent tous en logique négative (actif bas).

- CE : Chip Enable
- WE : Write Enable
- OE: Output Enable
- LB: 8-bits bas du mot de 16 bits.
- UB : 8bits hauts du mot de 16bits.

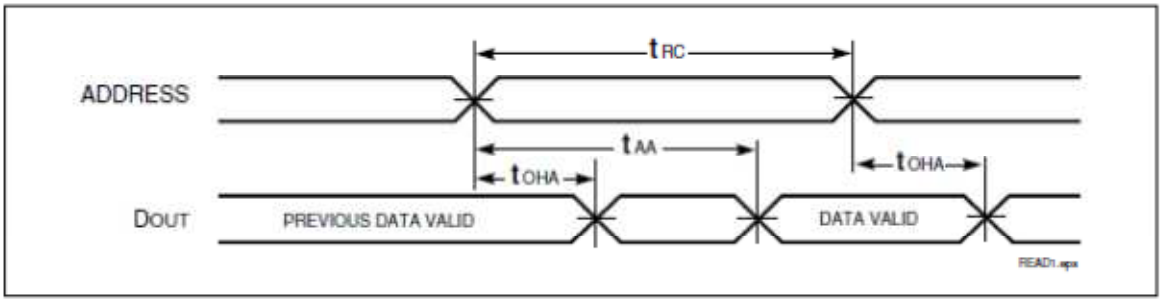
La table de vérité de la SRAM est la suivante :

						I/O PIN	
Mode	$\overline{WE}$	$\overline{CE}$	$\overline{OE}$	$\overline{LB}$	$\overline{UB}$	**I/O0-I/O7**	**I/O8-I/O15**
Not Selected	X	H	X	X	X	High-Z	High-Z
Output Disabled	H	L	H	X	X	High-Z	High-Z
	X	L	X	H	H	High-Z	High-Z
Read	H	L	L	L	H	DOUT	High-Z
	H	L	L	H	L	High-Z	DOUT
	H	L	L	L	L	DOUT	DOUT
Write	L	L	X	L	H	DIN	High-Z
	L	L	X	H	L	High-Z	DIN
	L	L	X	L	L	DIN	DIN

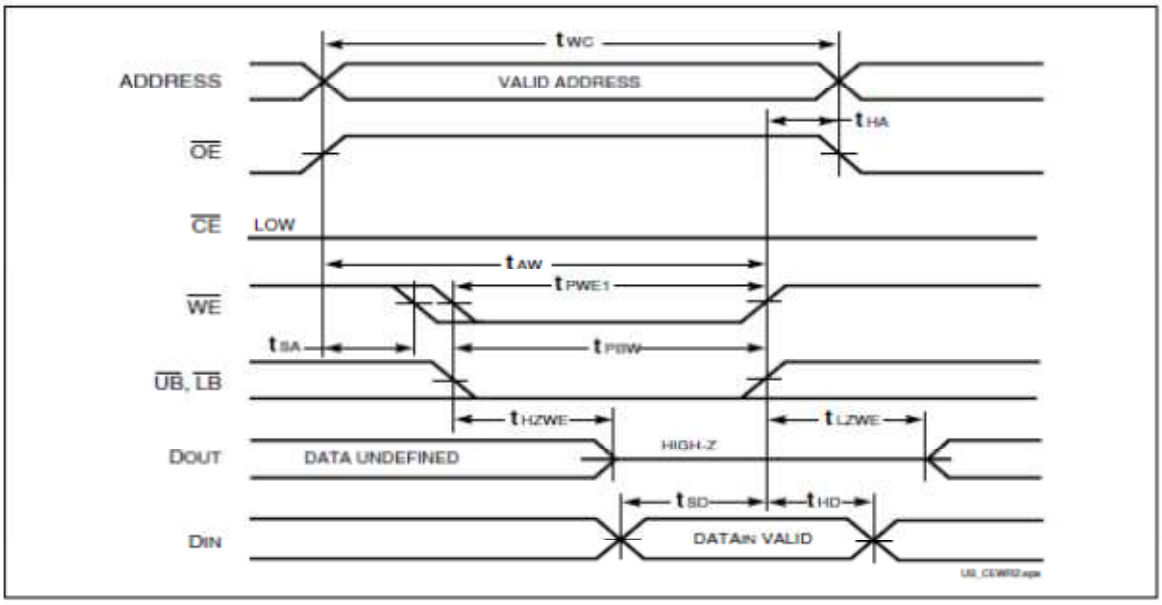
On y voit bien qu'il faut faire attention lorsqu' OE est bas, puisque le port de données bidirectionnel est en mode sortie, il faut donc que la sortie du port de donné du contrôleur VGA soit en mode haut-impédance. Avant de mettre au niveau bas le signal OE, on met la sortie du contrôleur VGA en mode haute-impédance.

Les chronogrammes suivants sont suggérés pour effectuer les lectures et écritures en mémoire :

- Lecture mémoire (CE = OE = '0', WE = '1', UB=LB='0')



- Écriture mémoire



Notez que tous les temps à respecter se retrouvent dans la datasheet. Essentiellement, la plus grande valeur de temps à respecter est de l'ordre de 10ns, et de nombreuses sont à 0ns. Lors de l'écriture d'un pixel en mémoire, vous devrez utiliser les signaux UB/LB afin de sélectionner l'octet adéquat pour l'écriture.

- **Pour le laboratoire, on vous fournit déjà le code VHDL complet pour réaliser le double buffer; assurez-vous de bien le comprendre pour l'évaluation car vous serez questionnés pour 0.5 point sur 5!**

Lorsque vous avez vérifié le fonctionnement du nouveau contrôleur VGA, modifiez la fonction ss_solar_system_iterate() de l'application SolarSystem3D pour en faire bon usage (**n'oubliez pas d'y enlever la ou les fonctions qui ne sont plus utiles**!). Faites toutes les modifications nécessaires dans le logiciel pour utiliser votre double buffer. Il est suggéré de vérifier d'abords le contrôleur VGA2 avec une petite application simple qui colore l'écran d'une couleur, par exemple.

- **En vous promenant dans votre système solaire, vous aurez probablement remarqué un petit problème avec le contrôleur VGA. En effet, il semble que les 2 dernières colonnes de droite se retrouvent à la gauche de l'écran. Ceci provient d'un décalage de 2 pixels par rapport à l'affichage; les pixels (0,0) et (0,1) sont inaccessibles. Corrigez ce petit problème dans le VHDL et assurez-vous du bon fonctionnement. N'oubliez pas de bien indiquer dans votre rapport vos explications.**

Instructions pour la Remise

Voir :

- ✓ plan de cours
- ✓ page d'informations sur Moodle

Vous devez remettre votre rapport ainsi que votre dossier de projet (contenant tous les fichiers projets !) compressé. Envoyez le tout sur la page Moodle du laboratoire, sur laquelle vous pourrez répondre à deux questions très courtes (facultatif, cela nous aidera à améliorer les laboratoires pour les années suivantes).

N'oubliez pas de sauvegarder votre travail et de nettoyer le disque temporaire D !

Il est accessible à tout le monde et est nettoyé toutes les nuits !

Grille d'évaluation

Ce laboratoire compte pour 6% de la session :

Partie 1 : 2/6

Partie 2 : 4/6